On the Use of Social Networking Groups for Automatic Configuration of Virtual Grid Environments

Pierre St Juste†, David Wolinsky†, Jiangyan Xu†, Michael Covington‡, Renato Figueiredo†
†Advanced Computing and Information Systems Lab, University of Florida, Gainesville, FL 32611
‡Corporate Technology Group, Intel Corporation, Hillsboro, OR 97124
ptony82@ufl.edu, davidiw@ufl.edu, jiangyan@ufl.edu, michael.j.covington@intel.com, renato@acis.ufl.edu

Abstract—**Social networking sites have enhanced the Web by providing online "communities" where users can easily create identities, establish relationships between identities, and share resources with one another across the Web. The availability of social networking APIs, such as OpenSocial or the Facebook platform, make it possible for third-party developers to tap into these social relationships and design socially-aware applications. In this paper, we describe how social networking groups can be used to help create virtual private computing clusters that consist of nodes spanning different administrative domains and organizations. To demonstrate the feasibility and usage of our approach, we developed a fully functional prototype implementation and present the quantitative results of our virtual private network. Our design integrates virtual machines, peer-to-peer virtual private networks, and the Facebook Platform API, to enable a user community in which we can easily deploy ad-hoc Condor resource pools that are managed through the Web-based Facebook group interface.**

I. Introduction

Online social networks have evolved into one of the most popular technologies of the so-called Web 2.0 era. Social networking sites are routinely used by tens of millions of users who interact with each other by sharing content, creating links, and maintaining relationships [1]. With the advent of social networking APIs, such as OpenSocial and Facebook Platform, developers can create smarter applications that leverage the social networking aspects of its users. Current socially-aware applications use existing links between users to cover a wide range of functionalities from sharing movie preferences to creating virtual private networks (VPNs) with friends [2]. Because user relationships are at the core of the establishment of communities and virtual organizations, it is important for the scientific arena to start exploring the advantages of integrating social networking in grid computing cyberinfrastructures.

Researchers can benefit from social networking concepts by utilizing social networking APIs to design simple and intuitive interfaces to grid computing environments. A fundamental goal of adapting grid computing environments to social networking is to to encourage collaboration among peers through this user-friendly interface. This paper demonstrates how to use social networking groups to facilitate the deployment of virtual clusters comprised of computing nodes located across different organizations. Our work is an extension to an ongoing research project aimed at creating easily deployable virtual grid environments [3]–[5]. This project uses pre-packaged virtual machines, virtual networking, and peer-to-peer overlay techniques to provide a scalable network of virtual resources suitable for grid computing. With our prototype, we create a virtual organization to deploy Condor pools on clusters of virtual machines distributed across different domains and subject to the networking constraints imposed by Network Address Translators (NATs).

In this paper, we present a system with the following characteristics: (1) a virtual machine environment that sandboxes the grid processing from the host system, (2) access to grid resources through a complete virtual operating system, (3) secure IP-level access to all virtual workstations through a virtual private network, (4) zero-configuration establishment of a virtual private cluster through a social networking interface.

More specifically, our prototype implementation allows a user to create a Facebook group that is used to manage access to a virtual cluster. By allowing users to join the Facebook group, the group creator in effect provides access to a private network of virtual workstations. Upon acquiring the virtual appliance image from our website, a social networking application, running within the virtual appliance, interacts with the user to authenticate and verify access to the Facebook group associated with the virtual cluster. Once verified, the virtual workstation automatically joins the virtual cluster and provide the user access to the grid resources.

There is ongoing research studying various aspects of online social networks, such as trust among users [6], user trends and relationships [1], and access control to

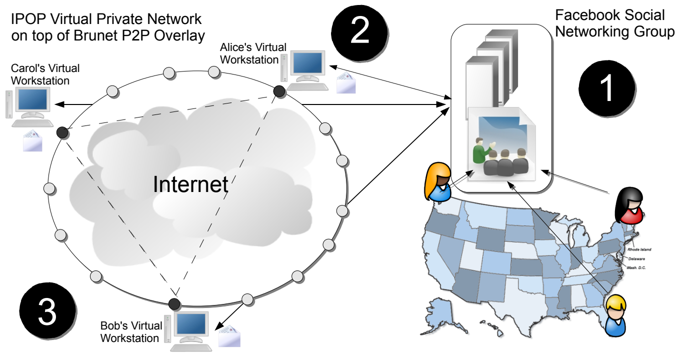

IPOP Virtual Private Network
on top of Brunet P2P Overlay
Alice's Virtual Workstation
Carol's Virtual Workstation
Internet
Bob's Virtual Workstation
Facebook Social Networking Group
1
2
3

Fig. 1. Virtual Private Network Overview: 1. Users from different networks form a Facebook group to share common research interests; 2. The social networking datastore obtains VPN settings and exchanges certificates; 3. The virtual workstations form a virtual private network on top of an existing Brunet peer-to-peer vpn overlay.

user content [7]. Yet, we believe that we have developed the first implementation that creates an automatic mapping between relationships established by a social networking group to end-to-end private tunnels in a virtual network. This is also the first social networking application to link distributed resources that are not controlled by the social networking site with identities that are generated and maintained by the site.

This paper addresses the problem of easily providing and managing secure access to grid resources. We demonstrate a social networking approach to creating and managing ad-hoc private virtual clusters enabled by a virtual private network. We eliminate the complexity of managing the virtual private network by combining its management with the maintenance of a social networking group. The paper also provides a novel approach for a certificate authority to handle certificate signing through a social networking paradigm. We also highlight the assumptions necessary of a social networking site to enable the feasibility of our approach.

The rest of this paper is organized as follows. We present the motivation and use case scenario for our approach in section II. Section III provides background information and assumptions of our key system components. In Section IV, we discuss the technical details of our prototype. We provide some analytical data and discuss the limitations of our implementations of Section V. Section VI covers related work and we concludes our paper in section VII.

II. Motivation

Deploying secure cross-organizational grid infrastructures is a non-trivial task. The complexity of managing these systems usually hinders collaboration among organizations. There is not an intuitive interface for managing network access across different organizations. The availability of social networking applications gives the opportunity to merge the group management interface of a social network with the creation and management of a peer-to-peer virtual private network.

We present the practicality of our work through a use case scenario. This discussion is on based on our Facebook/Condor prototype, but our approach can apply to other social networks, and grid-based applications.

Alice, Bob, and Carol are researchers at three different institutions who decide to share their spare CPU cycles among each other by using the Condor scheduler. Alice logs in to Facebook and creates a group called WideAreaResearchNetwork, and tells Bob, and Carol to join the group. Alice is notified through Facebook of their request to join the group, and confirms their identity by viewing their profile. Once their identify is verified, Alice adds them to the Facebook group. Viewing the requester's profile is one of the key contributions of a social networking site because it allows the administrator to identify a user through this *trusted* medium. Upon joining the group, Bob and Carol are instructed to download a virtual machine appliance encapsulating the O/S, middleware and applications that define a node in the virtual cluster, such as the Grid appliance [3]. Each of the researchers runs the appliance that uses the Facebook API along with the IPOP P2P overlay technology to set up an IPSec-secured virtual private network among the three virtual machines. Once the secure virtual LAN is created for these virtual workstations, the Condor job scheduler can run unmodified, as well as any other tools necessary for collaboration. Consequently, Alice, Bob, and Carol can securely share data, computing services, and CPU cycles.

III. Background

A. Social Networking Sites

Social networking sites serve as centralized datastores of user data and relationships. Application developers can use this existing data to create applications that facilitate connecting users to peers with similar interests. Most social networks allow users to create and manage groups to share information about common interests and resources. Upon joining a group, users allow the creator of the group access to their private data in order to identify the user. Group members can share data with other group members through the group's own data store. Social networking sites have provided APIs to store and access user data such as profile information, and application data. These APIs (OpenSocial, Facebook API) have led to the development of thousands p of social network applications. We leverage this functionality to alleviate the complexity of managing virtual private networks.

B. Assumptions of Social Networking Trust

Our work is based on various assumptions of trust in social networking sites. We assumed that the social networking backend is a trusted medium; hence, data associated with a particular user or group can be trusted. Another assumption is that social networking sites also provide accurate determination of user roles within a group (i.e. user A is a group administrator, user B is a group member). Through the use of a social networking datastore, users can exchange data by using the social links as a trusted out-of-band communication channel. Consequently, we have used the Facebook datastore to exchange certificate requests and retrievals among users and a certificate authority. These X.509 certificates are exchanged among peers to enable an IPSec-based virtual private network. Analyzing trust in social networks is beyond the scope of this paper.

Our virtual private network relies on the following requirements: (1) discovery of a Certificate Authority through a trusted medium, (2) acquisition of a CA public certificate through a trusted medium, (3) acquisition of a node certificate request through a trusted medium, (4) acquisition of CA signed certificate through a trusted medium. Our approach assumes that the social networking datastore can be used as the trusted medium to fulfill these requirements. The datastore thus facilitates the exchange of certificates between the CA and each member of the social networking group. The ability to link identity to data and our assumptions of trust in social networks is the main cornerstone of our approach.

C. Grid Appliance

The Grid Appliance project allows users to create wide-area networks of virtual workstations for desktop grid computing [3]. It uses virtual machines for easy deployment and sandboxing of the execution environment of the virtual workstations. These virtual workstations possess all-to-all IP connectivity through a virtual network technology called IPOP. IPOP stands for IP over P2P, and it uses a peer-to-peer overlay network to route IP traffic over the Internet, even when hosts are behind firewalls and NATs. By default, the grid appliance image runs the Condor job scheduler and each node becomes part of a Condor pool upon joining the virtual private network.

D. Virtual Private Network

The IPOP virtual network provides end-to-end security by encrypting the IP packets that are sent through the peer-to-peer overlay. Our implementation uses IPSec to encrypt IP packets sent between two nodes. Each virtual workstation needs a CA-signed certificate to communicate over the network. These certificates are used by IPSec to establish secure endpoint communication. The certificate exchange between a certificate authority and a valid endpoint must go through secure channels to guarantee privacy. Therefore, we use the social networking datastore as the trusted medium to securely obtain signed certificates from a certificate authority.

IV. Design Overview

We now elaborate on the technical details of our work. There are two main phases to consider in our implementation. The first phase is the configuration of the virtual private network to assure that the virtual workstations can properly connect. The second phase is the negotiation with the Certificate Authority to obtain a signed certificate. Each is thoroughly explained in the subsequent sections.

A. Joining an Independent Virtual Network

To properly simulate a virtual cluster, all-to-all IP connectivity is necessary. This is achieved through the IPOP (IP over P2P) virtual LAN. Through the use of a virtual network interface, we capture the Ethernet frames sent to a virtual NIC by the operating system. We extract the IP packet from these frames and route them over the Internet through the Brunet structured peer-to-peer overlay [5]. The Brunet P2P overlay is scalable to large numbers of nodes and can support multiple IPOP virtual networks. IPOP nodes are partitioned by a network identifier, and each identifier contains DHCP configuration stored in Brunet's distributed hashtable. Hence, nodes with the same IPOP network ID are part of the same virtual network and a local DHCP server automatically configures the virtual network interface [8].

In the current Grid Appliance, the creator of the virtual cluster distributes a virtual floppy image through a trusted channel (e.g. through an authenticated Web portal). The floppy image contains an IPOP network ID and DHCP configuration. Our approach uses the social networking group datastore as a trusted medium to share the configurations necessary for an IPOP node

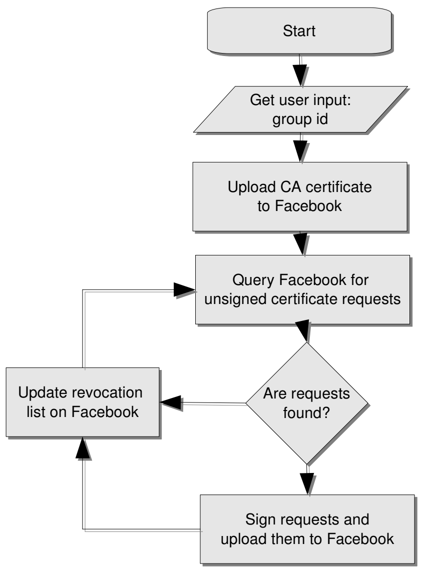

Start
Get user input: group id
Upload CA certificate to Facebook
Query Facebook for unsigned certificate requests
Are requests found?
Update revocation list on Facebook
Sign requests and upload them to Facebook

Fig. 2. Flowchart for social networking application in CA-mode. In CA-mode, the social networking application performs three main tasks: (1) publish the CA's certificate through the group's datastore, (2) check Facebook for certificate requests and sign certificates, and (3) publish and update the revocation list through the group's datastore.

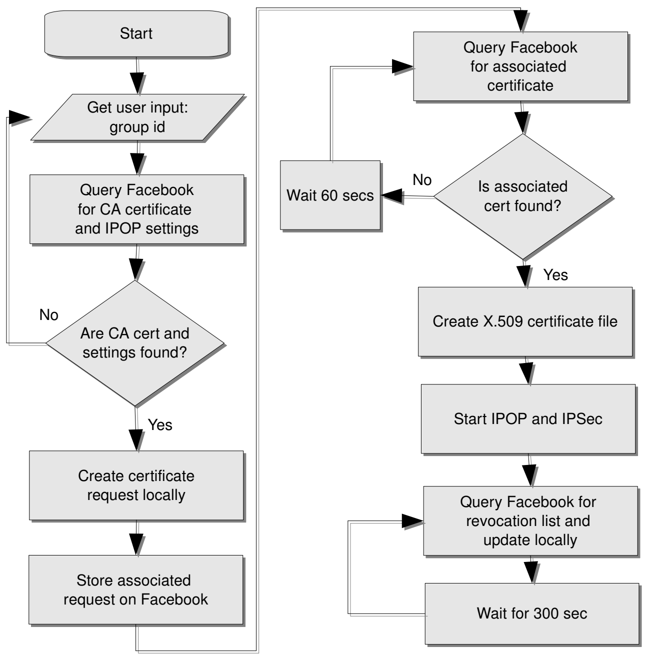

Start
Get user input: group id
Query Facebook for CA certificate and IPOP settings
No
Are CA cert and settings found?
Yes
Create certificate request locally
Store associated request on Facebook
Query Facebook for associated certificate
Wait 60 secs
No
Is associated cert found?
Yes
Create X.509 certificate file
Start IPOP and IPSec
Query Facebook for revocation list and update locally
Wait for 300 sec

Fig. 3. Flowchart for social networking application in client mode. In client mode, the social networking application performs the following tasks: (1) get the CA certificate and IPOP settings for the user-specified group, (2) create and publish a certificate request through Facebook, (3) poll Facebook for certificate for request and create certificate file, (4) start IPOP and IPSec, and (5) synchronize revocation list through the group's datastore.

to join the appropriate virtual network. These network configurations can only be retrieved by members of the social networking group. We developed a social networking application that runs within the virtual appliance. The application authenticates the user and extracts the network settings from the social networking group datastore. IPOP software uses these settings to join the virtual network associated with the current user's Facebook group. Users can join these virtual clusters by becoming a member of a virtual cluster group, and providing the group's name through the social networking application's web interface shown in Figure 5. The social networking application uses the user information to acquire the virtual network configurations and connect to the group's virtual private network.

B. Securing the Virtual Network through IPSec and Facebook

Our implementation uses IPSec to provide end-to-end security between virtual workstations. Therefore, each workstation is required to have a CA-signed X.509 certificate in order to securely communicate over the network. In traditional security models, various approaches are used to acquire a CA-signed X.509 certificate. Some organizations require a phone call to identify and acknowledge a certificate request, while others may find it satisfactory to do so over email or by requiring registration through a portal. In our system, we use the Facebook group datastore to exchange requests and certificates between the certificate authority and the group members. Once again, this is based on the assumption that the social networking datastore is a securely accessed and trusted backend. Figures 2 and 3 describe the process used in requesting and signing certificates which is explained in the following discussion.

1) Facebook Associations: In the Facebook API, application content is stored as data objects. Each data object has a unique identifier, and the creator of the object can associate it with his/her own identity or another data object. Our social networking application uses three types of pre-defined associations: (1) a request association that links a request to a user identity, (2) a certificate association that binds a certificate to a request or a CA certificate to a group, and (3) a revocation association to indicate invalid certificates to the group. The group creator, who acts as the certificate authority, is the only user capable of creating certificate and revocation associations. This restriction guarantees that certificates and revocation lists can only come from the certificate authority. Hence, a user makes a certificate request through Facebook by creating a data object containing an identifier (e.g. user Facebook id), and the request information. The user then associates this request data object to his/her identity. The CA also publishes certificates in a similar manner, but uses a certificate association to link certificate objects to request objects.

2) FQL:Facebook Query Language: Facebook has developed an SQL-like query language called FQL. With FQL, the Facebook database can be accessed more efficiently by allowing multiple API calls to be combined into one FQL call. This is important because Facebook limits the number of API calls to about 15 calls per minute. Here are some examples of the various FQL queries used by our social networking application: This FQL query returns the CA certificate:

```
SELECT id, data
FROM SecureGridNet.ipopdata
WHERE _id IN (SELECT obj_id
       FROM SecureGridNet.certificate
       WHERE gid = groupid)
```

This FQL query returns a list of certificate request ids:

```
SELECT obj_id
FROM SecureGridNet.request
WHERE usr_id IN (SELECT uid
                 FROM group_member
                 WHERE gid = groupid)
```

3) Getting CA Certificate from Facebook: In our security model, the group creator has the option to run the social networking application in CA-mode. In CA-mode, the social networking application publishes a previously created certificate and makes a certificate association between the certificate data object to the group id. The social networking API ensures that only the group creator can make certificate associations.

4) Making Certificate Requests through Facebook: In order to make a certificate request, each virtual workstation runs the social networking application in client mode. After acquiring the user-specified group name, the application is able to obtain the CA certificate and IPOP settings associated to that group. The social networking application creates a request data object containing that request, and associates this object to his/her Facebook user id. Subsequently, the social networking application checks Facebook periodically for a certificate associated to the request object.

5) Getting Certificate Requests from Facebook: After publishing the CA certificate, the CA-mode social networking application polls Facebook periodically for a list of request object IDs associated with its current members. This is done efficiently through the single FQL query shown above. In CA-mode, the application maintains a local list of signed request identifiers. Each time the list of request identifiers is fetched, it is compared to the local list to check for unsigned requests. A list of unsigned requests is generated, then the social networking application does another FQL query to obtain these certificate requests. Once these requests are acquired, a certicate is generated and signed by the CA.

6) Storing Signed Certificates through Facebook: Having signed the certificates, the social networking application stores them on Facebook by creating certificate data objects. Each certificate data object is then associated to its corresponding request data object. This association is used by the client application to properly retrieve the signed certificate corresponding to its certificate request.

7) Acquiring Signed Certificates from Facebook: As mentioned earlier, after creating the certificate request, the client application periodically checks Facebook for a certificate. An FQL query looks up the Facebook database for a certificate associate with the request data object identifier. If a certificate is returned, the social networking application creates an X.509 certificate file compatible with IPSec. Since only the group creator can create certificate associations, the retrieved certificate can be trusted through the constructs of a social network.

8) Handling Revocations through Facebook: There are some scenarios where certificates needs to be invalidated. One case is when a user no longer belongs to a group, either voluntarily or through removal by the group creator. In such a situation, the user's certificate needs to be invalidated to block access to other users in the virtual private network. Hence, the certificate authority maintains a revocation list. The revocation list is a data object associated to the group, and it is updated whenever a user leaves the social networking group. The client applications periodically updates their revocation list through Facebook. The revocation list ensures access control through the virtual network.

9) Putting it all Together: The acquisition of the certificate from the CA marks the end of second phase. Consequently, the social networking application can start the IPOP virtual network with the settings retrieved from the Facebook group. Once the virtual network is connected, the social networking application also applies

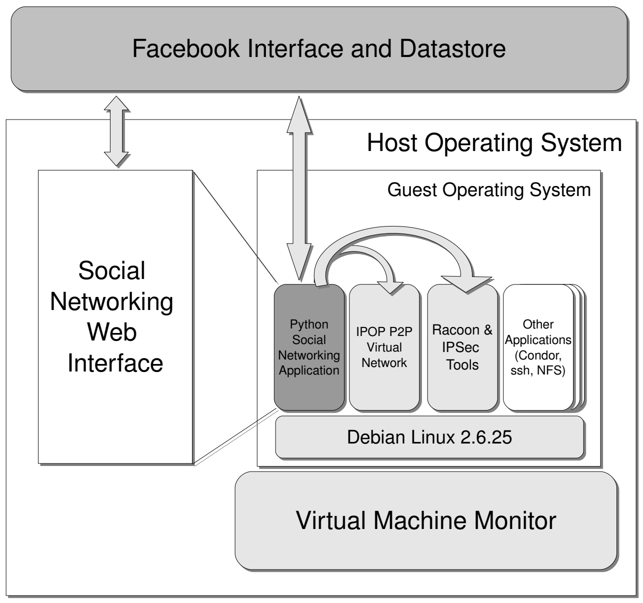

Facebook Interface and Datastore
Host Operating System
Guest Operating System
Social Networking Web Interface
Python Social Networking Application
IPOP P2P Virtual Network
Racoon & IPSec Tools
Other Applications (Condor, ssh, NFS)
Debian Linux 2.6.25
Virtual Machine Monitor

Fig. 4. Software Architecture. The social networking application runs inside a guest operating system running within a virtual machine. The user interacts with Facebook and the python-based social networking application. The social networking application is able to properly configure the IPOP P2P application and IPSec to enable the virtual private network.

IPSec on the virtual network to ensure privacy.

C. The Software Architecture

Our design builds on various open source software components. These components are visualized in Figure 4. Our virtual workstations run on top of a virtual machine monitor. In our experiments, we used VMware server 1.6 since it is freely available, but the appliance also works with the open-source software such as KVM and VirtualBox. The grid appliance runs Debian 4.0 Linux operating system with the 2.6.25 kernel. The virtual network interface is the universal tap device that comes with the Linux kernel. The IPOP peer-to-peer virtual network software is written in C# and runs on the Mono .NET Framework for Linux version 1.9.1.

The social networking application is responsible for configuring IPOP and IPSec to create a virtual private network. The application was developed in Python and runs on its 2.4 engine. We used the Python-Django version 0.96 web framework which provides the web interface to the user [9]. The open source PyFacebook API was used to communicate with Facebook social networking datastore [10].

To secure the network, we used the Racoon and Debian-based IPSec tools to apply IPSec encryption to the virtual network. We configured IPSec to use X.509 certificates for the creation of secure communication channels. For CA certificates, requests, and signing, we used the OpenSSL utilities. The social networking application managed the certificate handling process. Hence the application created certificate requests, signed certificates, and created the revocation list using the OpenSSL tools. The information from the social networking profile such as name, location, and city, was used to create the certificate requests. Overall, the integration of the social networking API facilitates the cumbersome tasks of handling and managing X.509 certificates.

V. Experiments and Analysis

We conducted various experiments to evaluate the complexity of deploying our virtual grid infrastructure. We also took some network measurements to analyze the performance of our virtual private network. We used the Condor job scheduler to run a BLAST gene sequence similarity search tool, which is a workload representative of academic research. Our infrastructure can potentially help scientists share and run simulations securely on virtual resources that are deployed across different organizations. Our main goal is to demonstrate the possible applications of our proposed grid cyberinfrastructure in the research environment.

Our experimental setup is comprised of five virtual machine Grid Appliances located in three different networks. These virtual machines are owned by two users, a group creator and a group member. Our system allows for multiple virtual machines on a single Facebook user account. Hence, the group creator ran three nodes: one node ran as the certificate authority, the other as a Condor manager, and the final node as a regular client Condor node. Another user account ran the additional two nodes in client mode.

A. Ease of Deployment

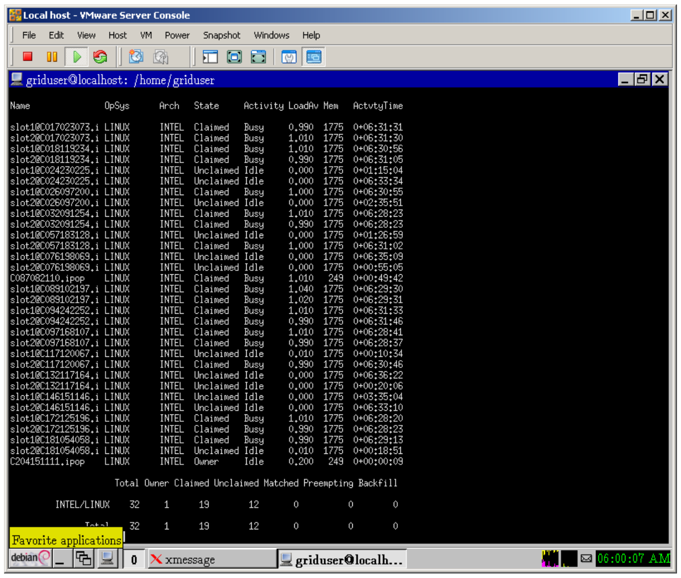
Deploying the virtual cluster was simple, it required zero-configuration, and it had almost no management overhead. The user's main tasks are: joining the Facebook group, installing a virtual machine monitor, and running the virtual machine image [11]. Once the virtual appliance is booted, the user navigates to http://apps.facebook.com/securegridnet/ from their browser. Figure 6 shows the web interface which allows the users to choose which group to associate with the virtual appliance as well as the mode of operation for the appliance. For our experiment, the virtual appliance could run in three modes relating to the Condor scheduler. In server mode, the virtual machine runs the Condor manager. The other two modes are client mode where the virtual machine can submit and run Condor jobs, and worker mode which can only run jobs. After the user makes their selection, the virtual appliance makes the certificate request through the Facebook datastore. This request is retrieved from Facebook by the CA, signed and stored back on Facebook. The acquisition of a signed certificate from the CA through the social network took less than three minutes. Upon receiving the signed certificate, the IPOP virtual network and IPSec is configured and started, a step which took less than a minute. The Condor job scheduler running on the client node, took less than five minutes to properly join the Condor pool. In a normal non-virtualized environment, joining a Condor pool might only take a few seconds because the Condor manager IP address is known a priori. In the Grid Appliance, a Condor-manager discovery process based on the P2P networks' DHT may take minutes for a Condor node to identify a Condor manager to connect with. Overall, it took less than ten minutes for a client nodes to securely obtain a signed certificate, configure its virtual private network, and join a Condor-managed pool of resources.

TABLE I
BANDWIDTH MEASUREMENTS

Protocol	Without IPSec	With IPSec
TCP	51.5 Mbps	33.0 Mbps
UDP	45.4 Mbps	32.0 Mbps

B. Network Performance

To quantify the overhead of our IPOP P2P virtual network and IPSec, we measured the throughput between two virtual machines within the same LAN. We took Iperf network measurements between two virtual machines in the same 2GB switched Ethernet network. Each virtual machine ran on a dual-core 2.33 GHz machine with 8GB of RAM. The virtual machines had 256MB RAM allocated to each of them. The measurements are shown in Table I. It is important to note that various sources of overhead beyond IPSec exist in our proposed approach. There is virtualized I/O overhead associated with virtual machines which can cause performance degradation. When we tested our IPOP virtual network software on unvirtualized physical hosts, we were able to achieve throughput above 200 Mbps. Since our deployment focuses on a wide area environment, a network bandwidth above 30 Mbps was sufficient for



```
Name               OpSys      Arch   State     Activity LoadAv Mem   ActvtyTime

slot1@C017023073.i LINUX      INTEL  Claimed   Busy     0.990  1775  0+06:31:31
slot2@C017023073.i LINUX      INTEL  Claimed   Busy     1.010  1775  0+06:31:30
slot1@C018119234.i LINUX      INTEL  Claimed   Busy     1.010  1775  0+06:30:56
slot2@C018119234.i LINUX      INTEL  Claimed   Busy     0.990  1775  0+06:31:05
slot1@C024230225.i LINUX      INTEL  Unclaimed Idle     0.000  1775  0+01:15:04
slot2@C024230225.i LINUX      INTEL  Unclaimed Idle     0.000  1775  0+06:33:34
slot1@C026097200.i LINUX      INTEL  Claimed   Busy     1.000  1775  0+06:30:55
slot2@C026097200.i LINUX      INTEL  Unclaimed Idle     0.000  1775  0+02:35:51
slot1@C032091254.i LINUX      INTEL  Claimed   Busy     1.010  1775  0+06:28:23
slot2@C032091254.i LINUX      INTEL  Claimed   Busy     0.990  1775  0+06:28:23
slot1@C057183128.i LINUX      INTEL  Unclaimed Idle     0.000  1775  0+01:26:59
slot2@C057183128.i LINUX      INTEL  Claimed   Busy     1.000  1775  0+06:31:02
slot1@C076198069.i LINUX      INTEL  Unclaimed Idle     0.000  1775  0+06:35:09
slot2@C076198069.i LINUX      INTEL  Unclaimed Idle     0.000  1775  0+00:55:05
C087082110.ipop    LINUX      INTEL  Claimed   Busy     1.010   249  0+00:49:42
slot1@C089102197.i LINUX      INTEL  Claimed   Busy     1.040  1775  0+06:29:30
slot2@C089102197.i LINUX      INTEL  Claimed   Busy     1.020  1775  0+06:29:31
slot1@C094242252.i LINUX      INTEL  Claimed   Busy     1.010  1775  0+06:31:33
slot2@C094242252.i LINUX      INTEL  Claimed   Busy     0.990  1775  0+06:31:46
slot1@C097168107.i LINUX      INTEL  Claimed   Busy     1.010  1775  0+06:28:41
slot2@C097168107.i LINUX      INTEL  Claimed   Busy     0.990  1775  0+06:28:37
slot1@C117120067.i LINUX      INTEL  Unclaimed Idle     0.010  1775  0+00:10:34
slot2@C117120067.i LINUX      INTEL  Claimed   Busy     0.990  1775  0+06:30:46
slot1@C132117164.i LINUX      INTEL  Unclaimed Idle     0.000  1775  0+06:36:22
slot2@C132117164.i LINUX      INTEL  Unclaimed Idle     0.000  1775  0+00:20:06
slot1@C146151146.i LINUX      INTEL  Unclaimed Idle     0.000  1775  0+03:35:04
slot2@C146151146.i LINUX      INTEL  Unclaimed Idle     0.000  1775  0+06:33:10
slot1@C172125196.i LINUX      INTEL  Claimed   Busy     1.010  1775  0+06:28:20
slot2@C172125196.i LINUX      INTEL  Claimed   Busy     0.990  1775  0+06:28:23
slot1@C181054058.i LINUX      INTEL  Claimed   Busy     0.990  1775  0+06:29:13
slot2@C181054058.i LINUX      INTEL  Unclaimed Idle     0.010  1775  0+00:18:51
C204151111.ipop    LINUX      INTEL  Owner     Idle     0.200   249  0+00:00:09

                     Total Owner Claimed Unclaimed Matched Preempting Backfill

         INTEL/LINUX    32     1      19        12       0          0        0

               Total    32     1      19        12       0          0        0
```

Fig. 5. Virtual Machine running in Client mode. This virtual machine provides the user with local desktop access to a virtual cluster. Users can submit Condor jobs as well as other tools necessary for collaboration such as NFS over a virtual private network.

our experiments. Overall, our results demonstrated that our virtual private network was able to accommodate the bandwidth requirements of a scientific grid computing task.

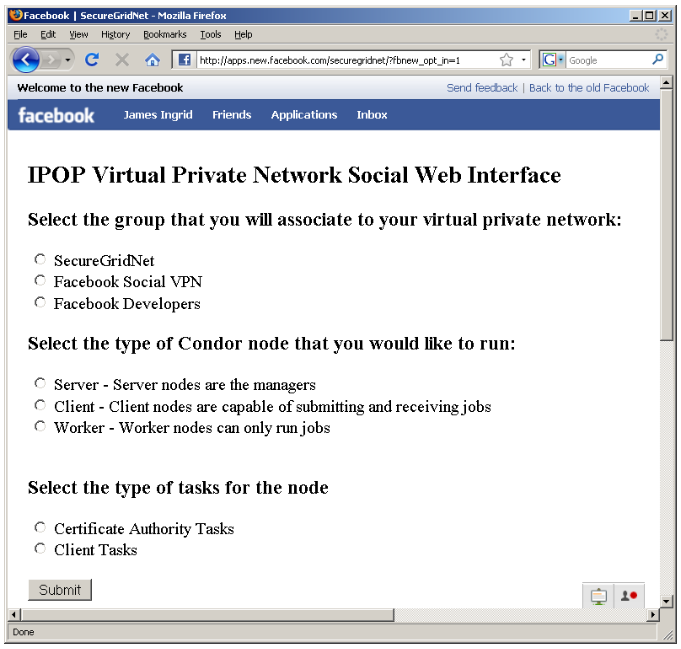
C. Performing Grid Tasks

We scheduled a BLAST job on our virtual grid environment to analyze the practicality of our approach. The Condor scheduler ran the job on each of the virtual machines and the results were sent to originating submit node. The job scheduler operated unmodified, behaving normally as if it was running on an physical LAN. The successful completion of our batch job proved that we can deploy unmodified applications that facilitates collaboration among researchers in different organizations.

D. The Limitations of Social Networks

In general, our approach can work with different social networks if they provide a datastore and APIs. We are currently designing an OpenSocial implementation that will allow us to use other social networks such as Orkut, MySpace, LinkedIn, and others. Throughout our paper, we have made the assumption that a social networking infrastructure is our trusted backend, but there are various limitations with the current Facebook infrastructure. The Facebook API is accessed through a REST server; therefore, all API function calls are HTTP GET or POST requests. Currently, Facebook does not provide a secure socket HTTP interface (HTTPS) to its API calls except for the Facebook login process. Since the communication channel is not secure with Facebook, it is not suitable as a trusted backend because it may be subject to man-in-the-middle attacks. Other social networking sites such as MySpace, Orkut, and LinkedIn, which support the OpenSocial protocol, have also not provided secure HTTP access to their resources. We believe that these issues will be addressed by social networking infrastructures as their API implementations mature.

The granularity of Facebook access control is also another issue. Even though there are various ways to provide access control to regular Facebook data, Facebook applications have access to all of user data that have added the application to their profile. Henceforth, application data on Facebook is global to all of the users of the application. Any user of the application can read data objects of other users if they can obtain the object's id. But only the creator of a data object is allowed to modify that data object. Despite the Facebook limitations, it is valid to mention that it is feasible and practical to use the social networking paradigm to provide a secure datastore for users in a collaborative community.


Facebook | SecureGridNet - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://apps.new.facebook.com/securegridnet/?fbnew_opt_in=1
Google
Welcome to the new Facebook
Send feedback | Back to the old Facebook
facebook
James Ingrid
Friends
Applications
Inbox
Submit
Done

IPOP Virtual Private Network Social Web Interface

Select the group that you will associate to your virtual private network:

- SecureGridNet
- Facebook Social VPN
- Facebook Developers

Select the type of Condor node that you would like to run:

- Server - Server nodes are the managers
- Client - Client nodes are capable of submitting and receiving jobs
- Worker - Worker nodes can only run jobs

Select the type of tasks for the node

- Certificate Authority Tasks
- Client Tasks

Fig. 6. Social Networking Application Web Interface. The user runs the virtual appliance locally, then goes to http://apps.facebook.com/securegridnet/. The application requests the user to select a Facebook group and a node type. Once the user submits a selection, the social networking application automatically configures the virtual appliance and joins the virtual private network.

VI. Related Works

The popularity of social networks and other Web 2.0 technologies has attracted much attention in academia. Mislove et. al [12] began exploring the advantages of social networks in Internet search. Their work demonstrated how Internet search can be enhanced by also searching the visited pages of friends in a social network. Although their work did not use a specific web-based social networking site, research projects have studied the characteristics of Facebook applications [13].

Recently, Curry et. al [14] presented work on how business enterprises can benefit from Web 2.0 technologies such as social networks and cloud computing by providing an intuitive environment for employees to access enterprise resources. Their implementation used Facebook to allow employees access to legacy applications running on a cloud-computing-based virtual appliances through a Facebook web-gateway. The main focus of this work is to create a better user experience for new employees in an enterprise, and easier access to tools within an organization. This work differs from our approach because it does not address the issue of cross-domain access to heterogeneous resources.

Other works have also talked about using social networks to establish secure channels. Authenticatr [15] uses social links for authenticated out-of-band communication to enable secure communication channels. Their approach advocates a middle layer in various applications in the same operating environment can use for peer discovery and cryptographic key exchanges. Their concepts are quite similar to our implementation, but they focus mainly on providing a socket-like messaging mechanism able to connect to many social networking sites and applications simultaneously.

Collaborative systems such as Grids provide efficient and scalable access to distributed computing capabilities and enable seamless resource sharing between users and platforms. This heterogeneous distribution of resources, and the various modes of collaborations that exist in the system require scalable, flexible, and fine-grained access control to protect both sensitive data and shared computing resources.

Originally in some Grid systems, each resource provider (RP) used a grid-mapfile to map external resource consumers (RC) to local identities and define their permissions. With dynamic user participations and resource sharing, this approach is not scalable.

The Community Authorization Service (CAS) [16] is a centralized approach, in which a CAS server maintains the access control policies and policy enforcement is managed on the CAS side. This approach allows resource providers to delegate some of the authority for maintaining fine-grained access control policies to communities, while still maintaining ultimate control over their resources. Although this approach solves the scalability problem, it lacks flexibility for ad-hoc collaborations. Also, CAS lacks flexibility to support a new resource provider which has not established a trust relationship with the CAS.

In contrast with solutions that involve centralized authorization, the Virtual Organization Membership Service (VOMS) [17] describes an approach in which each resource provider has a set of local policies. To access a shared resource, the user provides an attribute certificate issued from a virtual organization (VO) to identify user attributes, such as role, group name, and capabilities. By moving the policy decision point (PDP) from a centralized server to each RP's local site, VOMS can solve the scalability problem with the grid-mapfile and the flexibility of CAS, but it cannot support collaborations without a well-established infrastructure since it still requires a centralized attribute authority. Further, since an attribute in VOMS only includes role and group information in a VO, some policies cannot be implemented, such as user-level and VO-level delegation, and context-based authorization. That is, a user only can gain permissions from a VO administrator.

PRIMA [18] is a privilege management system which supports ad-hoc collaborations and permission delegation. To submit a request to an RP, a user provides a set of attributes which define the privileges of the user, such as file access permissions, user quota, network access, etc. Using these attributes and local policies, the RP assigns permissions to the user. A shortcoming with this approach is that, in a dynamic collaborative environment, the privileges of a user may change according to the resource consuming status in an RP, or some constraints with other concurrent jobs running in the RP. Therefore the pre-issued privilege attributes in PRIMA cannot support this dynamic and in-time permission assignments.

Zhang et al. [19] have proposed a usage control (UCON)-based security framework for collaborative applications that presents an architecture to support attribute mutability and decision continuity by leveraging a hybrid approach of attribute acquisitions and event-based updates.

Given the relatively recent gain in popularity for social networking, there has been very little work on access control frameworks that leverage the social networking infrastructure to improve resource protections or the usability of policy specification.

Carminati et al. [20] have presented an access control model for web-based social networks, where policies are expressed as constraints on the type, depth, and trust level of existing relationships. Their approach proposes a decentralized system architecture for access control enforcement, based on the interaction of two agents: the central node of the network, which stores and manages certificates specified by users, and a set of peripheral nodes, in charge of storing access rules and performing access control.

"Protection" and "security" in social networking-related research often focuses on the protection of resources and data. However, the relationships between users are also sensitive and need protection. Carminati et al. [7] have proposed a strategy that exploits cryptographic techniques to selectively disseminate information concerning relationships across a social network.

Extending this work, Domingo-Ferrer [21] has demonstrated a method that uses public-key cryptography to reduce the overhead caused by private relationships and minimizes the requirements for a centralized server.

Various additional Web 2.0 standards have encouraged the push for socially-aware systems. OpenSocial [22] is an open standard that propsoses a common interface for building social applications across different social networking sites. OpenID [23] aims at providing a distributed framework for digital identity. OpenID providers such as Yahoo.com, WordPress.com, LiveJournal.com, allows systems to identify users without requiring users to create specific accounts on these systems. OpenSocial uses OAuth [24], which is an open standard that defines how infrastructures such as social networks, can provide limited access to user resources without divulging user credentials.

VII. Conclusion

Social networking sites have gained tremendous popularity in the Web 2.0 era. With social networking APIs, developers can create a wide range of applications which makes it easy to stay connected with peers. Grid computing and social network share a common goal of resource sharing in an efficient, flexible, and secure manner. Thus, our approach integrates these two concepts to facilitate the creation of collaborative virtual grids.

Our work focused on the creation and management of a virtual private network that enables the deployment of a virtual cluster with nodes in different organizations. We demonstrated the simplicity, and practicality of using this virtual infrastructure as a gateway to virtual grid resources. Our network measurements and experiments confirmed that our virtual network was able to handle a common scientific computing task.

We were able to use the Facebook Platform API as a prototype social networking backend. We assumed that Facebook was a trusted backend able to bind identities and access control to stored and retrieved data. The current Facebook Platform does not possess all of the requirements to classify as a trusted backend. Despite these limitations, scientists can benefit by investigating the potential advantages of developing social-network-enabled grid infrastructures.

VIII. Acknowledgements

This research is sponsored by the National Science Foundation under grant IIP-0758596, the South East Alliance for Graduate Education and the Professoriate, the Florida Education Fund under the McKnight Doctoral Program. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the sponsors.

References

[1] A. Mislove, M. Marcon, K. P. Gummadi, P. Druschel, and B. Bhattacharjee, "Measurement and analysis of online social networks," in *Proceedings of the 5th ACM/USENIX Internet Measurement Conference (IMC'07)*, October 2007.

[2] R. Figueiredo, O. Boykin, P. S. Juste, and D. Wolinsky, "Social vpns: Integrating overlay and social networks for seamless p2p networking," in *IEEE Workshop on Collaborative Peer-to-Peer Systems (COPS'08)*, June 2008.

[3] D. I. Wolinsky, A. Agrawal, P. O. Boykin, J. Davis, A. Ganguly, V. Paramygin, P. Sheng, and R. J. Figueiredo, "On the design of virtual machine sandboxes for distributed computing in wide area overlays of virtual workstations," in *First Workshop on Virtualization Technologies in Distributed Computing (VTDC)*, November 2006.

[4] A. Ganguly, A. Agrawal, P. Boykin, and R. Figueiredo, "Wow: Self-organizing wide area overlay networks of virtual workstations," *High Performance Distributed Computing, 2006 15th IEEE International Symposium on*, pp. 30–42, 0-0 2006.

[5] ——, "Ip over p2p: enabling self-configuring virtual ip networks for grid computing," *Parallel and Distributed Processing Symposium, 2006. IPDPS 2006. 20th International*, pp. 10 pp.–, April 2006.

[6] J. Golbeck and J. Hendler, "Inferring binary trust relationships in web-based social networks," *ACM Trans. Interet Technol.*, vol. 6, no. 4, pp. 497–529, 2006.

[7] B. Carminati, E. Ferrari, and A. Perego, "Rule-based access control for social networks." in *OTM Workshops (2)*, ser. Lecture Notes in Computer Science, R. Meersman, Z. Tari, and P. Herrero, Eds., vol. 4278. Springer, 2006, pp. 1734–1744. [Online]. Available: http://dblp.uni-trier.de/db/conf/otm/otm2006-w2.html/CarminatiFP06

[8] A. Ganguly, D. Wolinsky, P. Boykin, and R. Figueiredo, "Decentralized dynamic host configuration in wide-area overlays of virtual workstations," *Parallel and Distributed Processing Symposium, 2007. IPDPS 2007. 21th International*, pp. 8 pp.–, March 2007.

[9] The django python web framework. [Online]. Available: http://www.djangoproject.com/

[10] The python facebook api. [Online]. Available: http://code.google.com/p/pyfacebook/

[11] The grid-appliance project. [Online]. Available: http://www.grid-appliance.org/

[12] A. Mislove, K. P. Gummadi, and P. Druschel, "Exploiting social networks for internet search," in *Proceedings of the 5th Workshop on Hot Topics in Networks (HotNets'06)*, November 2006.

[13] M. Gjoka, M. Sirivianos, A. Markopoulou, and X. Yang, "Poking facebook: Characterization of osn applications," in *Proceedings of the 1st ACM SIGCOMM Workshop on Social Networks (WOSN'08)*, August 2008.

[14] R. Curry, C. Kiddle, N. Markatchev, R. Simmonds, T. Tan, and M. Arlitt, "Facebook meets the virtualized enterprise," in *12th IEEE International EDOC Conference (EDOC'08)*, September 2008.

[15] A. Ramachandran and N. Feamster, "Authenticated out-of-band communication over social links," in *Proceedings of the 1st ACM SIGCOMM Workshop on Social Networks (WOSN'08)*, August 2008.

[16] L. Pearlman, V. Welch, I. Foster, C. Kesselman, and S. Tuecke, "A community authorization service for group collaboration," *Third International Workshop on Policies for Distributed Systems and Networks*, pp. 50–59, June 2002.

[17] R. Alfieria, R. Cecchinib, V. Ciaschinic, L. dell'Agnellod, A. Frohnere, K. Lorentey, and F. Spatarog, "From gridmap-file to VOMS: Managing authorization in a grid environment," in *Future Generation Computer Systems*, vol. 21, no. 4, April 2005, pp. 549–558.

[18] M. Lorch, D. Adams, D. Kafura, M. Koneni, A. Rathi, and S. Shah, "The prima system for privilege management, authorization and enforcement in grid environments," *Grid Computing, 2003. Proceedings. Fourth International Workshop on*, pp. 109–116, Nov. 2003.

[19] X. Zhang, M. Nakae, M. J. Covington, and R. Sandhu, "Toward a usage-based security framework for collaborative computing systems," *ACM Trans. Inf. Syst. Secur.*, vol. 11, no. 1, pp. 1–36, 2008.

[20] B. Carminati, E. Ferrari, and A. Perego, "Private relationships in social networks." in *ICDE Workshops*. IEEE Computer Society, 2007, pp. 163–171. [Online]. Available: http://dblp.uni-trier.de/db/conf/icde/icdew2007.html/CarminatiFP07

[21] J. Domingo-Ferrer, "A public-key protocol for social networks with private relationships," in *Lecture Notes in Computer Science*, vol. 4617 (Modeling Decisions for Artificial Intelligence-MDAI'2007), no. ISSN: 0302-9743, August 2007, pp. 373–379.

[22] Opensocial. [Online]. Available: http://code.google.com/apis/opensocial/

[23] Openid. [Online]. Available: http://openid.net/

[24] Oauth. [Online]. Available: http://oauth.net/