The George Washington University
Request for Comments: 8000
Category: Experimental

A. Maloney
December 2017

Open Social Graph Protocol

Status of This Memo

This memo is in a draft state meant for evaluation and is open to suggestions. Distribution of this memo is limited to CSCI 6431/4331 students and professors.

Abstract

This document specifies an experimental open social graph protocol for the Internet Community. It does not specify an Internet standard of any kind. This protocol would allow for communication of social graph objects between a host and client machine, facilitating communication between social network platforms that implement this protocol. This is an application layer protocol and relies on Transmission Control Protocol (TCP) and Internet Protocol (IP) for transportation between client and server.

Table of Contents

1. Introduction 2
2. The OSGP Model 3
2.1 Basic Structure 3
2.2 OSGP Terminology 3
2.2.1 Social Graph 3
2.2.2 Social Graph Objects 3
2.2.3 Social Graph Object Connections 4
2.2.4 Posts 4
2.3 Example Graph 4
3. OSGP Procedures 5
3.1 Create Graph 5
3.2 Request Graph..... 5
3.3 Send Notification 5
3.4 Request Posts 5
3.5 OSGP Server Notification Handler 6
4. OSGP Specifications 6
4.1 OSGP Commands 6
4.2 OSGP Replies 7
5. Security Considerations 7
6. Author’s Address 7

1. Introduction

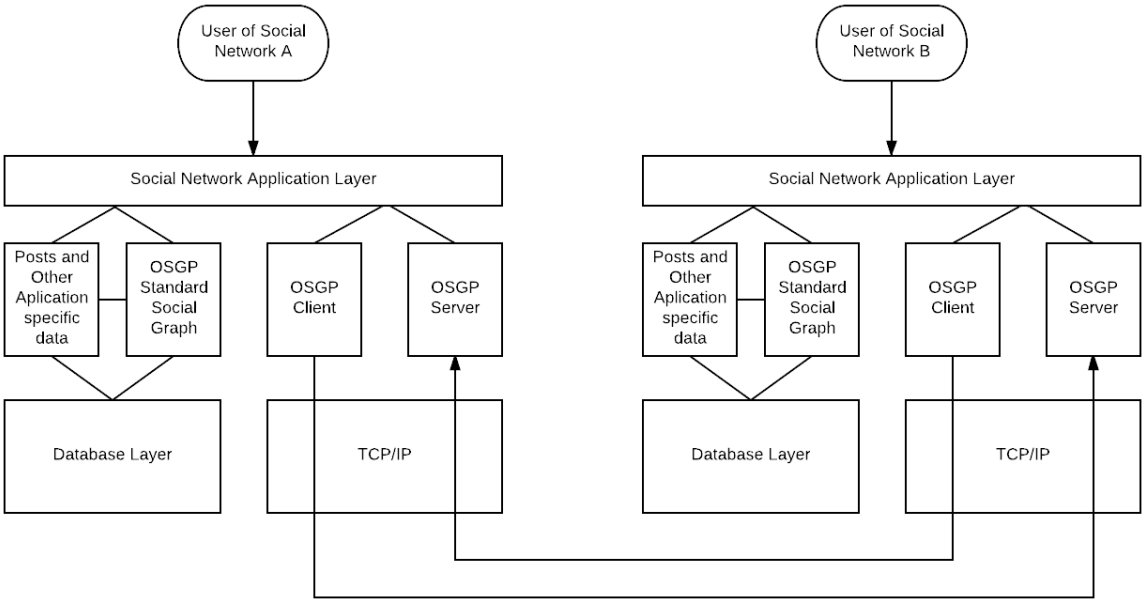
The Objective of the application layer protocol Open Social Graph Protocol (OSGP) is to transfer social graph objects reliably and efficiently.

The motivation for designing such a protocol is to allow for social networking in an open ecosystem. Current social networks completely own a user's social graph and are able to do as they please with it, selling the data or using it to target users with specific ads.

With a defined protocol for the communication of social graph objects between a client and server, social networking platforms could be built independently of the social graph. The social networking platform would implement OSGP and the user would retain ownership of their social graph. Additionally, this would allow for communication between social networks that implement OSGP, a feature that is impossible in the current closed ecosystem.

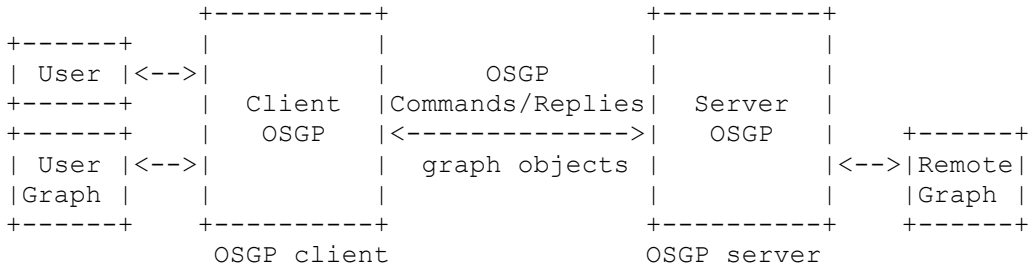
The intention of this protocol, though possible, is not for individual users to operate their own locally stored graphs, but for independent social networks to implement this protocol. Much in the same way that users do not directly operate their own SMTP servers, but instead use independent clients such as Gmail and Yahoo to interact with and receive mail.

The following diagram describes how this protocol fits into the application stack for a social network, such that it would allow users of different networks to communicate.



2. The OSGP Model

2.1 Basic Structure



When a client would like to request a graph object or notify a remote server it establishes a two-way transmission channel to an OSGP server which listens for incoming connections on port 5005. Messages will be passed between client and server via the transport layer transmission control protocol (TCP) [RFC 793] which provides reliable data transfer which is critical to ensure objects are correctly transmitted as well as the network layer Internet Protocol (IP) [RFC 791].

The Client first presents the server with the operation code it wishes to perform on the graph. The server will issue a response code denoting the client's capability of performing such an operation, if permitted the server will then responded with the excepted data as defined by server responses via TCP/IP.

In the case of a client object request to a remote server, the client will provide the unique ID of the object they are requesting and the server will respond with a response code, and if permitted the objects will be transmitted to the client.

2.2 OSGP Terminology

2.2.1 Social Graph

A graph data structure is defined by a set of nodes and edges between those nodes. A Social Graph is a collection of Graph Objects and connections that represents a user's "social network". This object is stored managed by the application that implements OSGP and is able to be modified and transmitted through OSGP commands.

Applications that implement this protocol do not necessarily need the same social graph implementation but the base structure must be defined as described in this section.

2.2.2 Social Graph Objects

These objects make are the nodes of the social graph. Objects must contain the following information: a globally unique ID defined by

the user name and home domain of that user (same concept as an email address), the type of object so the receiver knows what data to expect, details as defined by the type, and a list of post objects.

An example object:

```
      Object ID     : maloneya@myface
      Type          : User
      Details       : {name: Alex Maloney, age:21}
      Posts         : {234523,23443,33234}
```

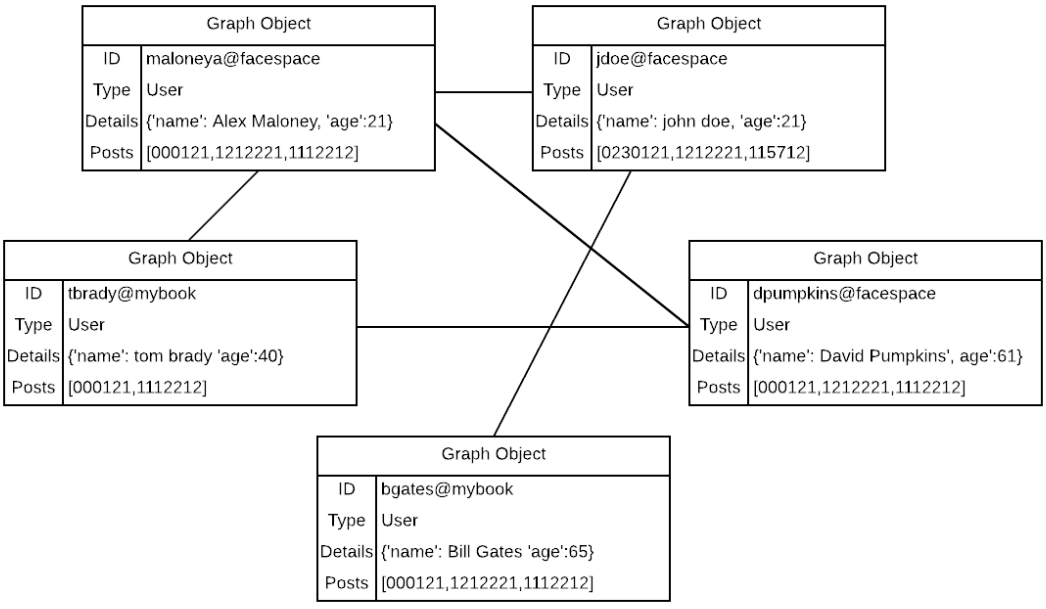
2.2.3 Social Graph Object Connections

Connections between graph objects can represent various things on the application layer depending upon implementation. Connections are defined by an object pair (i.e., (Jane@myface, jill@myface)). The storage schema for connections is out of the scope of this protocol; however, all connections must be transmitted along with their respective graph objects in order to maintain the graph through transmission.

2.2.4 Posts

Posts allow for the representation of content in a social network and are not actually graph objects themselves, but are instead referenced from graph objects. Each object maintains a list of its posts. Their storage concerns are beyond the scope of this application however the server must communicate its type to the client upon transmission. The list of posts stored on a user object can be updated via pull request from client to the server that owns the graph object. Or the owner of an object can update and add to their posts via application layer logic.

2.3 Example Graph



3. OSGP Procedures

All communication procedures will attempt to initiate server connections at TCP port 5005. The Server determines client permissions to perform operations by checking if the requesting user on the client side is connected to the remote user in the server's social graph. If a client is unable to find a server it will wait and attempt to make a connection again.

3.1 Create Graph

This is an initialization procedure that must be taken upon first use. This procedure will set up the root node of the social graph. This procedure can also be implemented in the application that relies upon this protocol, if the application would prefer to manage the graph.

3.2 Request Graph

This procedure will attempt to make a connection with a remote server and request a graph object. The client will send "request_graph" to inform the server it is requesting a graph. If it receives an ok from the server, it will send the same of the user whose graph it is requesting and will provide the id of the requesting user. If the user is not found at the remote server, the server will respond with a not found error and if the requesting user is not permitted the server will respond with an unauthorized response. Otherwise the server will transmit the graph to the client.

3.3 Send Notification

This procedure will send a notification to an OSGP server by first sending the server "notify". If the client makes a successful connection it will transmit the name of the remote user it is notifying, the type of notification and the id of the sender. If the server cannot find the user to notify it will respond to the client with a not found error code. If the server notification handler cannot process this type of notification it will respond to the client with a not implemented error code. Otherwise it will push the notification up to the application and notify the client of successful transmission.

3.4 Fetch Posts

This procedure will attempt to make a connection to the remote server and pass it the "request posts" operation. If a connection is established the client provides the remote graph object id and the id of the requester. If the user is not found or the requester is not permitted the server will respond with an error. Otherwise all post data is transmitted to client.

3.5 OSGP Server Notification Handler

In addition to listening for requests the server must implement a notification handler. The notifications the server is capable of processing can be expanded depending upon the needs of the social network application. The only notifications the server must implement are "connection added" and "connection removed". These notifications inform a server when a client has created or removed connections between nodes owned by this server. If server receives a notification it doesn't know how to handle it responds to the client with a not implemented error code.

4. OSGP Specifications

4.1 OSGP Commands

- CREATE_GRAPH (OWNNER ID, OWNER DETIALS)
  This is a local procedure that must be performed to initialize the graph.

- REQUEST_GRAPH(REMOTE_DOMAIN, REMOTE_USER, REQUESTING_USER)

```
Client: request_graph ---->
                                          <---- 200 : Server
Client: remote_user   ---->
Client: requester_id  ---->

                      <---- if user not found 404 : Server
                      <----  if not permitted 401 : Server
                                <---- user graph: Server
```

- REQUEST_POSTS(REMOTE_DOMAIN, REMOTE_USER, REQUESTING_USER)

```
Client: request_posts ---->
                                          <---- 200 : Server
Client: remote_user   ---->
Client: requester_id  ---->

                      <---- if user not found 404 : Server
                      <----  if not permitted 401 : Server
                            <---- user's posts    : Server
```

- SEND_NOTIFICATION(REMOTE_DOMAIN, REMOTE_USER,
  NOTIFICATION REQUESTING_USER)

```
Client: notify ---->
                                          <---- 200 : Server
Client: remote_user   ---->
Client: notification  ---->
Client: Sender ID     ---->

                      <---- if user not found 404 : Server
          <----    if notification not impled 401 : Server
                                   <----   else 200: Server
```

4.2 OSGP Replies

The following coding scheme will be used for response codes, leaving room for future expansion.

1xx Informational Reponses
2xx Success Codes
4xx Client Errors
5xx Server Errors

200 – Command OK
400 – Bad Request: the server cannot correctly process the request
401 – Client not authorized
404 – User not found
500 – Internal server error
501 – Not implemented

5. Security Considerations

This protocol with provide security through a secure socket layer (SSL) [RFC 6101] transmission of all graph objects. This is a natural fit for a client server protocol as it is able to provided privacy between two communicating applications with reliable data transfer.

Additionally, each user could have a public and private key that they use to sign graph operations. This would allow OSGP servers to verify that incoming requests and notifications are being performed by the user that imitated the connection.

6. Author's Address

Alexander N. Maloney
The George Washington University
800 21st Street NW
Washington, DC 20052 USA

Email: maloneya@gwu.edu