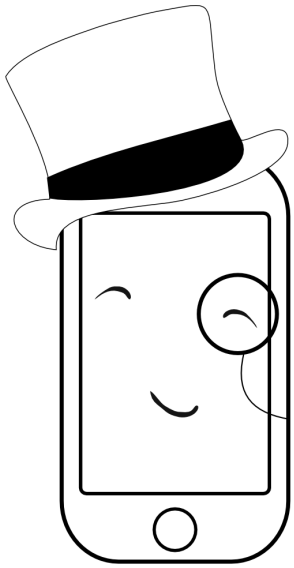

Society of Machines

1 Introduction

Despite being researched for over 15 years, Internet of Things hasn't really arrived. In general, computing devices do not seamlessly connect to one another and share their information. It takes considerable effort by a application developer to make two machines cooperate on the same network (much less on different networks) and in many of these applications security is lacking. This proposal describes an architecture to help overcome the obstacles preventing widespread use of device to device communication and finally deliver on the promise of the Internet of Things.

The Society of Machines library enables machine communication and distributed applications by forming secure/reliable peer to peer networks with discovery, tag based searching, network permissions and proxying for NAT transversal. The library is built on the secure variant of the ZMQ library, uses a Chord implementation for peer to peer network formation/maintenance and SQLite databases as part of its distributed storage. A overlay network is formed for each set of permissions, with select members optionally joining a larger network to allow for groups to be easily found by devices that have the required privileges.

From a developer stand point, Society of Machines allows an application to securely publish message streams and be assured that (only) the correct parties hear it regardless of what network the application is located on. Information streams and servers in the peer to peer network are tagged/ searchable, meaning new applications can find information without knowing anything about the architecture of applications providing the information.

2 Applications

With these capabilities, some tasks that are normally very hard become much easier. For instance, one potential use is a fleet of small mobile robots for an automated delivery service on NC State University campus. Orders would be placed via a web/mobile interface and then various robots (potentially of several different types) would be assigned to complete the actions necessary to complete the delivery. With the SoM library, one or more servers can provide the front end and be provisioned with permissions sufficient to allow them to send orders to delivery robots (via appropriately tagged request/reply interfaces on the robots). The various types of robots can be tracked simply by publishing a GPS data stream with a particular tag using a known packet format (such as a Libprotobuf template). Upon being granted sufficient privileges on the SoM network, a new server can easily get a list of all robots and their information simply by performing a tag search for appropriately tagged request/reply interfaces and GPS streams.

If desired, information on the network can also be made public by subscribing to streams (forming an aggregator) and then posting the information on a website and/or RSS stream. This makes it possible to build front ends to

the network with minimal knowledge of the devices/applications providing the data. For instance, a GUI front end that displays the position of all robots in the network could be made by either subscribing to robots' published streams with the appropriate tag or sending requests for GPS information to a preexisting aggregator (also found by a tag search). Alternatively, the a server could perform processing and publish its results on the SoM network. This would allow for the formation of multiple layers of aggregators performing increasingly high level analysis to create large and complex systems.

A more immediate application is use of Society of Machines to enable home and industrial automation. A SoM enabled application can publish information by defining the permissions (who should be able to see the stream on which networks) and the tags associated with the stream/server. If the networks they are publishing either have one member registered to the global SoM peer to peer network or the library has an IP address of one of the nodes in the network, the application can just publish and the library will take care of all of the other details. This means that there can be a clean division between applications/devices that provide sensor information and those that process data to provide meaningful information. If the library becomes popular, it is quite possible that we would see the emergence of an SoM enabled app store that would allow people to find or buy software that would process the data their devices are collecting in new and meaningful ways.

A side effect of implementing home automation using the Society of Machines library is that many of the services which have been provided by streaming user data to the cloud could be performed using solely the user's devices. This would allow the user to regain control of their data and possibly prevent unscrupulous use by third parties. It also allows for users to be able to access their network when not at home via NAT transversal and secure authentication/encryption (making it simple to create applications that allow remote monitoring on mobile devices such as smart phones).

Another capability enabled by the Society of Machines library is granting other users permissions that are limited in scope and duration. If a user at some point acquires a lawn mowing robot, it is unlikely that they will have enough lawn to keep the robot busy all the time. With a SoM interface, it is quite possible that we will see scenarios where a user grants their neighbor control over a specific part of their network (such as the request interface and GPS stream of a lawn mowing robot) for a specific time in a specific region. In this specific instance, the neighbor could be granted permissions to access the mower robot every Tuesday from 2 pm to 4 pm while the robot was in the GPS range that included its resting place and the neighbor's yard. Other possible arrangements are job spot market for robots where a person's software agents look at the robots available in the current region, examine their capabilities/asking rate and arrange to have robotic services (such as lawn mowing) performed at minimal cost. It would also be possible for people to donate hours of their robotic agents to charitable causes, such as street cleaning or delivery for a food bank. These varied applications are all possible without changes to the underlying Society of Machines library.

3 Architecture

The previous discussion has focused on what the proposed library can do and enable. What follows is centered on how the proposed library operates. The ZMQ networking library has recently added official support for its encrypted variant (CurveMQ). The CurveMQ variant provides reliable publisher/subscriber, one to one connections and request reply messaging with strong encryption and authentication. The ZMQ version can be used for unencrypted interprocess communication on a machine. This forms the basis of information streams and servers for Society of Machines. The SQLite database is being used to enable tag based searching. There is not a suitable Chord library available to form relatively secure Chord networks, so that part of the library is being developed based on the Chord specification. Permissions are implemented based on a permission fragments concept which helps hide the full permission set an application has access to.

Permission fragments define 4 mandatory fields (network ID, node encryption key, permission key and permission value) and 2 optional ones (permission start time and permission end time) and are signed by one or more certificate authorities (generally the owner of the network acts as the certificate authority). The information in a permission fragment corresponds closely to the following: Certificate authority A says to give this node status X at level Y from start time to end time. This could be as narrow as Give neighbor Jone's SoM network mowerAccessPermission at level 5 from 2 pm to 4 pm next Tuesday and as broad as Give node NSA access to everything forever. When a request or query is sent to a SoM node, generally one or more permission fragments are sent with it to prove that the owner of the encryption key has sufficient rights. The SoM node can then choose whether or not to accept the fragment based on many factors (including who signed it). Likewise, the requester can choose exactly how much of their authority they wish to reveal by deciding which fragments to send. There are three types of nodes in Society of Machines. Servers are defined as nodes having a static and public IP address and permissions sufficient to be allow them to help with tag based searching and provide proxy services. In general, these would be back end servers with strong Internet connections and physical security. Peers are defined as nodes with static and public IP addresses that don't have sufficient permissions to provide tag based searching and proxy services. The public IP requirement can be relaxed if the servers and peers are only communicating with nodes on their network, but that isn't really the best use of the library. Clients are nodes that don't have static or public IP addresses and use proxy services to access the network.

Upon starting, a server checks what permission fragments it has available and which Society of Machine networks they correspond to. It then searches for IP addresses of servers on that SoM network that either the administrator added or were stored in previous sessions and tries to contact them to join the network (verifying they have proper permissions and showing them the permissions required to join as a server). If that is not sufficient to find all of the SoM networks it has been given access to, the server then contacts the global

SoM Chord network (which has several well known URLs/IP addresses) and searches for IP addresses of nodes associated with the required SoM networks. It joins any networks it finds in this manner. If there are still SoM networks that it has been unable to resolve, the server starts a new Chord network for each missing one and optionally publishes its presence on these networks to the global SoM Chord network.

When a server has entered a network, it is given a portion of the Chord address space and takes responsibility for answering queries about resources which have tags associated with that address space. The server can then publish information, publisher request servers and send requests to other nodes. The server also becomes available as a proxy to allow other machines to publish, act as servers and send requests.

Upon starting, a peer does the same steps as a server except for becoming part of the Chord network and acting as a proxy. It can search for resources, publish information, create servers and send requests. It does not, however, help with with storage and therefore will not get access to tags and information it does not have sufficient permissions to use. When a peer sends query to a server, the server filters from the reply all network resources that the peer has not shown sufficient permissions to access. This means what the peer can see on the network is restricted to what it should be able access.

Upon starting, a client does the same steps as a peer except for being able to directly publish/advertise resources. Since a client doesn't have a reliable and public IP, instead of directly making a CurveMQ socket for information publishing it instead asks a server to act as a proxy. If the server chooses to accept, the client forms a one to one connection with the server which it uses to send requests to publish resources and send information. When the client chooses to publish something, the server publishes a resource indicator that has all of the same tags but points to the server and indicates that they are acting as a proxy. Nodes sending requests then talk to the server and have the conversation forwarded back and forth with the client. If the client changes IP, the one to one connection is destroyed and after a delay (to allow for reconnection) the server will remove the outgoing streams and tags. The client can form a new one to one connection either with the same server (reconnecting) or a different one (republishing the tags).

4 Conclusion

The proposed architecture for the Society of Machines library would appear to offer the capabilities that are required for the use cases that have been discussed and is feasible to construct using existing libraries as a base. The functionality offered could enable the fulfillment of the potential inherent in the Internet of Things and allow for the construction of cooperative robotic services.