Secure Distributed
Open
Social Networks

Henry Story
Senior Staff Engineer
Semantic Web Evangelist
http://blogs.sun.com/bblfish
Sun Microsystem
Overview

1. Description of the Social Networking problem
2. Why this is no longer “somebody else's problem”: a hyper address book
3. The functioning of the Address Book
4. How to add distributed decentralized security
5. A final thought: how this changes the desktop paradigm
Too many Social Networks?

...are there too many web servers?
The Problem: data silos

- SN don't link up:
  - Information can't be moved easily (see: Data Portability the video)
  - Users have to create and maintain accounts on each SN they have friends on, or loose contacts

- Growing number of social networks (SN)
  - because there are a lot of $$$ to be made
  - because there are many needs
  - there will never be one SN to rule them all.
In early January 2008 Scoble, the developer who got blogging going at Microsoft, got thrown off Facebook for extracting information too aggressively from his social network on Facebook.

This is the Facebook who asked users for their gmail password to extract all their contacts from their email!

see his video
An (evolving) Social Graph

relates many different things
- people to information about them
  - name
  - address
  - phone number
- relations between people:
  - who knows who
  - who worked with who
- relations with external things
  - blogs
  - companies
Two social Networks

how can Tim and Henry link up?
Solution 1: minimal naïve approach

but within each SN queries are very limited:
e.g.: in Network A, nobody can query for Tim's address
Solution 2: copy some information

- how to copy the data? Data Portability? (DRY principle?)
- how to keep the relations up to date?! Twice as much work.
- queries still limited: what are the friends of Tim's friends?
Solution 3: copy all

- technically impossible: does not scale as networks grow in size and number:
  - how to keep information up to date?
  - amount of synchronization grows exponentially
- politically impossible: S.N. are very protective of their data + privacy issues + oligopoly issues
The pull to one network

Due to Metcalf's law: the larger the network the more valuable it becomes.

But why does it have to be in one database?
Because each database has its own LOCAL POINTER mechanism, just like every Java virtual Machine has a local pointer mechanism. You cannot easily point from one JVM/DB into another.

What if we had one big world wide database? we would need universal names for things. URIs?
The Solution: linking across social networks requires a global namespace
The Solution: a closer look

- objects and documents have URLs
- Relations also have URLs: foaf:knows, foaf:name

→ The Self Describing Web
A hyperdata Address Book

The So(m)mer Address Book

Description

The hyperdata Address Book project being developed here is meant to be the equivalent for foaf that BlogReaders are for RSS. It is a specialised Semantic Web browser that follows foaf documents around the web, building a distributed open social network. It is also a foaf editor, which you can use to publish your foaf files to an ftp/scp server.

With foaf you no longer need to worry about being locked off your Social Network. You can own your web server/domain name or rent it from some trustworthy provider that allows you to publish anything you want. You could even put it on your home server linked to the internet. You can link to anyone whenever their server is and anyone can link to you. You can even publish any information you want. RDF is very flexible.

This is in my opinion the big issue for 2006. For a list of blog posts on the subject see the addressbook tagged feed.

Download the latest version (26 August 2006). Clicking that link should start Java Web Start. If not please let me know what browser/soa combination you were using.

Building

Download, build and run the Address Book by typing the following commands (leave the password to the svn checkout blank). This currently requires Java 5 and 6.

```bash
> svn checkout https://sommer.dev.java.net/svn/sommer/trunk sommer --username guest
> cd sommer/misc/AddressBook/
> ant run
```

Note: the ant run no longer works from the command line since adding Nasa Wind. You will need to build using Netbeans 6.1 until I find out how to fix this javaWebStart build problem... (I am investigating)

If you want to try the code out on different virtual machines then instead of ant run execute (note: no longer works for same reason as previous note above)

```bash
> ant jar
> java -jar dist/AddressBook.jar
```

Then you can just drag and drop the foaf file icon into the first column of the address book, and follow the distributed social network by pressing the space bar to get foaf files.

Participate
1. first launch of jnlp
2. drag and drop a foaf file url
3. click on the first name in the first column
4. explore the second column
5. press the space bar on the keyboard...
6. positioning with NASA's World Wind
7. Sun Intranet Foaf experiment
Foaf: Friend of a Friend

Syntax

Semantics
Advantages

- Open Social Network – no data silos
- Information about people is always up to date (an HTTP GET away)
  ➔ this could be used to keep up to date on where friends are
- It is easy to publish a foaf file: one click away
- Drag and drop friends
- Security: some ideas at the end of the talk
Two foaf files on the internet
Well, what we really have is

```
@prefix : <http://ongoing.org/tim.rdf#>
@prefix foaf: <http://xmlns.com/foaf/0.1/>
:tim a foaf:Person;
  foaf:name "Tim";
  foaf:workplaceHomepage <http://sun.com>;
  foaf:openid <http://ongoing.org>;
  foaf:knows <http://bblfish.net/people/henry/card#me> .

@prefix : <http://bblfish.net/people/henry/card#>
@prefix foaf: <http://xmlns.com/foaf/0.1/>
:me a foaf:Person;
  foaf:name "Henry";
  foaf:workplaceHomepage <http://sun.com>;
  foaf:phone <tel:+33.(0)164225952>;
  foaf:knows <http://ongoing.org/tim.rdf#tim>
```

Well, what we really have is in graph view
The graphs inside the Beatnik Database
Networked graphs: A merged view
SPARQL: semantic query lang

- PREFIX foaf: <http://xmlns.com/foaf/0.1/>
- SELECT ?p
- WHERE {
}
SPARQL construct query

PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

CONSTRUCT {
} WHERE {
    GRAPH ?g {
    }
    ?g :fetched-at ?date .
    FILTER { ?date < "2008-03-30"^^xsd:date }
}
Networked Graphs: SPARQL Rules

PREFIX owl: <http://www.w3.org/2002/07/owl#>

CONSTRUCT { ?b owl:sameAs ?a . }
WHERE {
    ?a owl:sameAs ?b .
    FILTER ( ! SAMETERM(?a , ?b) )
}

CONSTRUCT queries can also be thought of as rules. Here is the well known rule of symmetry of identity.

Simon Schenk's *Networked Graphs* can have a number of rules expressed as SPARQL CONSTRUCT queries, which works nicely with the Sesame semantic engine.
merging identities

PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

CONSTRUCT { ?a owl:sameAs ?b . }
WHERE {
  ?a foaf:homepage ?pg .
  FILTER ( ! SAMETERM (?a , ?b))
}
Security: 3 approaches

1. Simple Firewall based security
2. OpenId based Security
3. Even simpler SSL based security
Firewall protection
Protecting resources with OpenId
Protected Resources

:me a foaf:Person;
  foaf:name “Henry Story”;
  rdfs:seeAlso </protected/henry> .

</protected/henry> openid:login </openidAuth.cgi> .

- This is trying to say that in order to access the protected resource one needs to login with openid first.
- This is a sketch of such a vocabulary
OpenId continued
foaf+ssl: even simpler
Protected Resources

:me a foaf:Person;
  foaf:name "Henry Story";
  rdfs:seeAlso <https://.../protected/henry> .

• notice the seeAlso is now an https url
foaf+ssl: even simpler
X509 certificate

Certificate:
  Data:
    Version: 3 (0x2)
    Serial Number: 1 (0x1)
    Signature Algorithm: dsaWithSHA1
    Issuer: O=OpenPGP to X.509 Bridge, OU=RDFauth Test, CN=Henry Story <henry.story@bblfish.net>
  Validity
    Not After : Dec  6 21:49:50 2008 GMT
Subject: O=OpenPGP to X.509 Bridge, OU=RDFauth Test, CN=Henry Story <henry.story@bblfish.net>
  Subject Public Key Info:
    Public Key Algorithm: dsaEncryption
    DSA Public Key:
      pub:
        33:41:...
X509 certificate with id

X509v3 extensions:
  X509v3 Basic Constraints: critical
    CA:TRUE
  X509v3 Key Usage: critical
    Digital Signature, Non Repudiation, Key Encipherment, Key Agreement, Certificate Sign
  Netscape Cert Type:
    SSL Client, S/MIME
  X509v3 Subject Key Identifier:
  X509v3 Authority Key Identifier:

  X509v3 Subject Alternative Name:
    URI:http://bblfish.net/people/henry/card#me
  Signature Algorithm: dsaWithSHA1
      eb:8c:11:08:1c:aa:93:7d:71:01
Very Simple Authentication

Diagram:

1. Romeo SemWeb Client sends GET request to Juliette's Semantically enabled Web Server.
The Semantic Desktop

- hyper-
- mosaic
- beatnik semantic address book
- master/slave
- windows 95 explorer
- minitel
- windows 3.1 file manager
- island
- microsoft word
- LinkedIn
- email
some references

• Getting Started With RDF
• The Semantic Address Book web site