Personal Information Protocol

Michael G. Baumer
Allenmoosstr. 54
CH-8057 Zürich
mbaumer@vis.ethz.ch

June 28, 1999

Abstract

This paper documents the Personal Information Protocol (PIP) Suite. It describes the design considerations. Further the usage of the protocol is shown with a sample implementation of a PIP-Server and a PIP-Client.

Contents

Chapter 1

Introduction

The Internet has become a personal communication utility for many of us. But while it enables us to send messages using e-mail and to chat with poeple around the world we cannot easily find out if some user is currently using the internet and where he is located (i.e. in terms of computer addresses). Nor does it provide a standard way to request information about a user. We present in this paper a new protocol that tries to answer this request.

Protocols that address this problem exist. The two main categories of protocols differ in if they are centralized or not. We show in Chapter 2 how these protocols work and what their advantages and disadvantages are.

In Chapter 3 we discuss the goals a protocol for personal information should reach.

We then define the Personal Information Protocol in Chapter 4.

The usage of this protocol is shown with a sample implementation of a PIP-Server (Chapter 5) and a PIP-Client (Chapter 6).

Chapter 2

Existing Protocols

Several Protocols exist that try to address the problem. Only the oldest of these protocols has a decentralized approach.

2.1 Finger

2.2 Centralized Systems

Chapter 3

Goals

3.1 Protocol

- Groups of Friends/collegues/working/hobby etc. and world/other
- different groups, different available data
- automatically
- use exsiting infrastrucure, chat $->$ talk-protocol
- messages & file transfer: perhaps a new protocol is needed. Problem: What if user not online? Where get the messages stored? (use e-mail instead?). Storing messages on the server is not useful if we assume that e-mail address equal pip address.
- configurable finger output (with identd)
- Portability
- Open Protocol specification
- Extensibiltiy / Version numbering / Feature test
- Distributed environment: Information about a user is not stored centrally but at a server of his choice. (even better security).
- Changing working environment: easy switching of working station.
- Forwarding: like mail forwarding but information is transferred to the client which may update its database.
- Security: Identify people with public key infrastructure

3.2 Feature List

Transfer information about the following:

- Users current host, if possible the console host
- Personal Information (Birthday, Name, Homepage, gender, address, about, etc.)
- Talk/Chat: Yes/No, Protocol: talk/ICQ/costum

- File Transfer: ftp is probably not sufficient, for permission reasons
- Send message: Notice, e-mail, URL
- Announce status changes (logoff, logon, away, etc...)
- login time, idle time, “where”, what(?), JCPU (à la finger). Only visible to few poeple.
- login $< - >$ Real Name
- Allow super servers
- Make a addressbook server
- No central server necessary, but build one for poeple without access to a PIP-Server. Only store IP of user when he logs on and forward it to poeple requesting information. Perhaps store some little information like Full Name and Homepage.
- Use e-mail adress as Identifier: world-wide uniquness, distributed. But let the PIP-Server override the default e-mail address.
- Security: Send messages encrypted, if they go over the network.
- allow Key/Value Pairs to be added and changed remotely
- Multiple Languages
- xface ?

3.3 Client

- Client: Buddy list possible
- Client: Integrate other protocols like ICQ, finger v1.
- Client: Try to examine existing address-books.

  pine: .addressbook: Nick `TAB` Full Name `TAB` e-mail, where e-maillist = e-mail | '('e-mail {',' e-mail} ')'. e-mail is “xyz.com” or “X Y <xyz.com>”.
- Good default config file
- make shared protocol libraries
- check mailbox for mail from friends and grab it if the user wants to see it.
- Make new (small) finger client: only fetch user info and disconnect.

Chapter 4

Protocol Specification

- portnumber: finger 79/tcp
- Use the existing finger port. New PIP (aka finger-) daemon ist recognized by a ID string that is sent before the user has sent anything. This does not break the old finger client and it enables the new client to wait for this string. Perhaps we find another method...
- Try all host we know as finger v1 servers at client startup time.
- persistent tcp connections to other users
- UDP keep alive connections
- UPD or TCP connection for login-information, depending on the number of people on the other side.

4.1 Finger Protocol v1

Old Finger daemons provide a single request/response protocol, where the answer is in an arbitrary format. The protocol is specified in *rfc*. We may extract the login-name, the real name and the online status. But we have to be open about where the tokens are to find, or how the keys look like. Further multiple answers may be received. Best use of this protocol is, if we know the correct username, to provide the online status.

Chapter 5

Sample Server Implementation

server daemon listen on server port

in.fingerd: no fork necessary

semantics of gnu fingerd?

architecture: look up local users (/etc/passwd) and local logins. Then read from a database. fill this database with login/logout commands. A site fingerd may request the data from all hosts and login all poeple in the main finger-site. since clients have tcp-connections or semi-connected udp-connections to the server, logouts can be detected.

5.1 General Layout

5.2 Site Server

Like the GNU Finger suite we provide a way to have a “site server”.

Chapter 6

Sample Client Implementation