Requirements for Personalized User Environments in Telecommunications

Dr. Jens-Peter Redlich
Department of Computer Science
Humboldt University Berlin
redlich@informatik.hu-berlin.de

Dirk Balfanz
Department of Computer Science
Princeton University
balfanz@cs.princeton.edu

Abstract - "The network is the computer" is a vision currently coming true. That "computer", however, is characterized by a complexity, chaos, and abundance of information which makes it increasingly difficult for a single (let alone non-expert) user to efficiently utilize it. In this paper, we propose an architecture that allows users to have their own personal view of the network, and therefore make "the network their personal computer". In our architecture users can describe their Personal Working Environment (PWE) which will be instantiated on the terminal they are using regardless of the type of terminal equipment or their specific location within the network. This demands that resources in the respective vicinity of the user be considered for inclusion in his/her Personal Working Environment and that the terminal be flexible enough to present different incarnations of the same resources in a consistent manner to the user. We will describe the components needed for such an architecture and report on a prototype implementation.

Keywords: telecommunications, personalized working environment, mobile computing, network resource management

I. Introduction

Telecommunication networks are at the heart of today's emerging "information superhighways". The rapid development in this area allows global access to numerous kinds of resources (literature, software, or music data bases; on-line communication, interactive games etc.). The Internet is swiftly gaining momentum in a growing community of non-expert users. The current situation presents itself as follows:

- Resources that in the past have been available to a small number of users only are now globally available in a unified network (the World Wide Web). There are already tools to access these resources (e.g. FTP, Telnet etc.)
- Today, "the network is the computer." The user no longer sees the personal computer as a machine that provides computing power or other valuable resources by itself, but rather as a door to the network which has become to provide the resources the user is interested in.
- The number of available resources is exceeding the amount of information that a single user can possibly consume. Therefore, the net appears complex, chaotic, unstructured, and unmanageable to the user. Many visible resources do not permit the user to access them. Moreover, most of the resources potentially available to a user are of no interest to him/her, but still add to the apparent complexity of the network.
- The number of different types of resources is also constantly increasing (ASCII text, Postscript, RTF, HTML, X.500 entries, GIF, TIFF, JPEG pictures, WAV, MULAW, ADPCM sounds, JPEG/MPEG videos etc.). Tools like Netscape try to account for that (inline plug-ins) but can barely keep pace with the development.
- The speed of the increase of the aforementioned factors is accelerating, too. More powerful development platforms are being introduced that allow to build complex distributed applications (e.g. IN, TMN, TINA-C in telecommunications or CORBA, DCE in distributed programming). Soon, the user will be helplessly drowning in a sea of (seemingly) useless information and will not be able to find the (limited number) of resources he/she looks for.
- Especially mobile users are hurt by the complexity and tremendous heterogeneity that governs all the developments described above. When in a new environment, numerous details stand in the way of simply plugging the user's laptop into the network at the new location: Configuration of modems, FAX's, email clients etc. Furthermore, if the user doesn't travel with his/her own terminal (laptop), there is often no possibility of gaining access to the network at all.

In order to avoid negative implications of these developments and to foster acceptance of telecommunication technology, a stronger emphasis on individual user needs is necessary. Most prominently, this concerns the presentation of the network itself to the user. We believe that at least the following requirements should be met:

- Users must be able to describe the resources of interest to them. This way, the number of visible resources could be limited to a manageable amount. The resultant set of resources represents the user's personal view of the network (and re-establishes the notion of a "personal computer"). We shall call this personalized viewpoint the user's **Personal Working Environment (PWE)**. The user's PWE is the network representation that he/she is used to and that he/she therefore can use most efficiently.

- Users decide on their own which features of a specific resource are of interest to them and in what way a certain functionality should be provided. (Has for example a user access to a file server, he/she might want to neglect the fact that that host also serves as an HTTP server.)
- Regardless of where the users are and whether or not they use their own terminal, their PWE should be delivered to them in the same way. That means that for some abstract description of resources (e.g. "nearest printer", "local weather forecast") actual service providers have to be found, whereas other resources (e.g. "POP account: `user@mailhost.domain.edu`") will always stay the same and need merely be accessed. Some of the resources might not be available and must be emulated to establish the specified PWE (e.g. no printer might be available in the near vicinity but a fax machine. Low quality print-outs could be redirected to that fax machine.)
- Users need a verifiable and unique identity which is the same for all types of services. Today, this is not the case. When we want to call someone, we usually have to know where this person is and then dial the number of his/her *telephone*. If we do not get through, we might have to know this person's email address in order to leave him/her a message etc.

It turns out that in local area networks solutions exist for some of these problems:

- With Sun's NFS in place, users find themselves in their home directory regardless of which workstation in the LAN they use.
- Sun's NIS (former yellow pages) allows a user's identity to be know LAN-wide. Hence, authentication and authorization work across the (local area) network.
- Again, thanks to NFS, rudimentary user profiles (e.g. X start-up and/or configuration files) are available across the network and enable a consistent working environment (as long as the user logs into machines of the same kind).
- To some limited extend, services can even be dynamically "chosen" to reflect the capabilities of the actual terminal the user uses (e.g. `setenv PATH "/usr/local/`uname`/bin"`).

The solutions employed for local area networks cannot be used in a global telecommunications context, though. They rely too heavily on a central system administration and a small number of users/resources. Scaled to a global setup, if at all possible, they would be hopelessly inefficient and probably unreliable.

The remainder of this paper will be structured as follows. In chapter II we will introduce the components of a simple architecture that is to address the requirements derived above. Chapter III will then show how these components interact in order to provide a personalized view of the network which is independent of actual resources, and hence delivers the same services to the user regardless of his/her type of terminal or location.

In chapter IV we report on a prototype implementation of these concepts which is based on the World Wide Web. Closing, we discuss our results and present an outlook to the future.

II. What Do We Need?

In this section we will describe components that we believe to be necessary for personalized user environments in a global telecom infrastructure. They aim to give the user his/her unique view of the network and to turn it into his/her "personal computer". In chapter III. we will then go on and describe how the components to be introduced in this chapter interact and together provide a Personal Working Environment for the user.

A. User Profile Service

The User Profile Service (UPS) maintains a description of the user's abstract Personal Working Environment (PWE). A PWE is the user's personal view of the network. It will probably describe a subset of all available network resources (in order to make them available to the user) and configure appropriate local resources (e.g. enforce preferences regarding GUI layout etc.). The description is abstract because it may contain references to abstract network resources ("the nearest printer", "the local travel agencies" etc.). When logging into the network, the user will ask the UPS to fetch his/her (abstract) PWE, turn it into an instantiated PWE, and build his/her user environment according to the instantiated PWE.

The UPS can be implemented in a highly distributed fashion. No central UPS server is necessary, the abstract PWE may even be stored in the user's network terminal. It must be certain, though, that the user may access his/her abstract PWE at all times and from all points within the network.

1) Why an Abstract PWE?

Why do we need an abstract description of the user's personal working environment? The reason is that we want a consistent view of the network across changes of both the user and the network itself:

- The user and/or the terminal can change. The user can for example upgrade his/her terminal equipment so that it can provide better quality of service. Was the terminal before only able to display movies at a

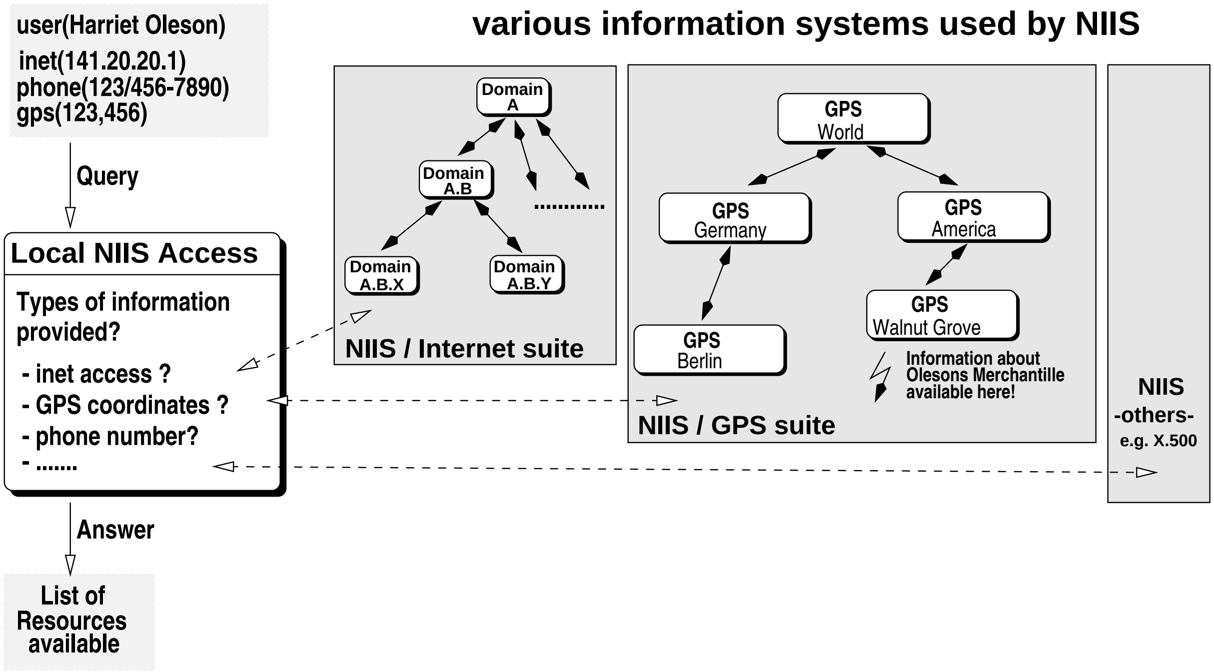

Fig. 1. The Network Infrastructure Information Service

rate of 10 fps, it may now be able to display movies at 25 fps. A single (abstract) specification of "highest possible frame rate" should allow the terminal to automatically adjust to hardware upgrades. To give another example, the user might be travelling around the world and access the network from different terminals which have different capabilities. Some might have printers attached to them, some not (and would therefore have to employ a, say, designated commercial printing service to meet print requests) etc. An abstract description of the PWE helps overcoming these heterogeneities.

- Also, the network can change. Again, if the user travels around and plugs his/her terminal into different network access points, the network will present itself differently to the terminal (and, hence, to the user): Well-known servers might not be available, the link might be slower than usual etc. Also, the network infrastructure itself constantly changes. Links become faster, the underlying transport protocols change etc.

An abstract PWE allows us to describe a user's working environment independently of available resources. Only for a concrete instantiation of this environment we choose from the set of available resources those that are needed to implement the various components of the personal working environment in the best possible way (considering the actual conditions at the time and place of instantiation).

B. Network Infrastructure Information Service

The Network Infrastructure Information Service (NIIS) maintains a repository of actual network resources. Supplied with information about where the user is at the moment, it will find out what resources are available in the vicinity of the user and produce a list of resources that he/she has access to. It will usually be queried by the PWE-trader (see chapter III.C) which is trying to instantiate a user's abstract PWE as delivered by the UPS.

Obviously, the set of all network resources is by far to huge to be handled by one server. Therefore, it is necessary to employ a distributed implementation of this service, which potentially knows about all resources on the network. The Internet Domain Name Service might serve as a role model: All NIIS servers are connected in a tree-like fashion and accept queries about all parts of the network. If a server finds that it cannot provide information about the requested region, it will pass the request on to its superordinated server, or, if one of its subordinated servers can provide the information, to that subordinated server. Figure 1 is to exemplify this strategy.

The NIIS can utilize various specialized location services that will allow to associate location information with available resources. As the location information can be given in various forms (e.g. GPS coordinates, IP address, phone number etc.) different specialized services are needed to map a specific location information to a list of available resources.

Notice that the NIIS only knows about static (or slowly changing) information in the network: It might for example know that in location X there is a printer but it does not know whether or not the printer is switched on or is out of paper. This is because the amount of information that is handled by a global Network Infrastructure Information Service is so vast that for performance reasons data will have to be cached (just

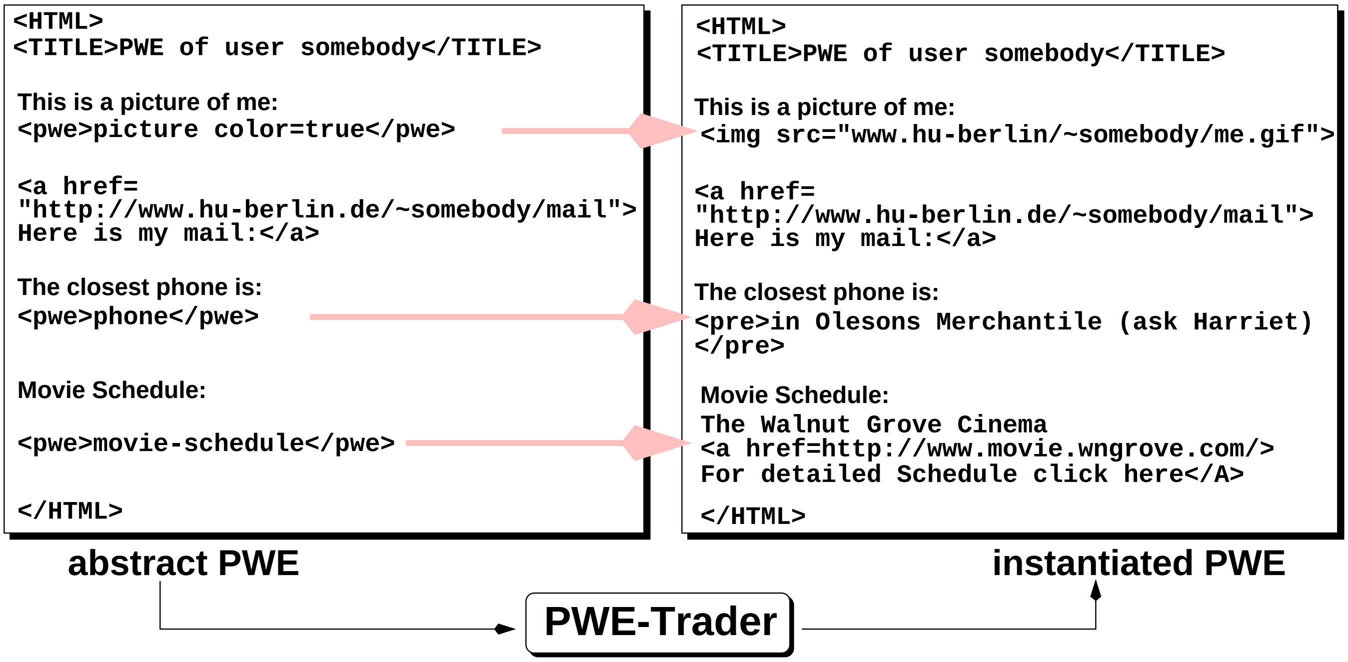

Fig. 2. How a PWE-Trader works

as in DNS). Caching highly dynamic data can not be done efficiently, though. For this reason another component, the PWE-Trader, is being introduced.

C. PWE-Trader

The PWE-Trader is the component that converts the abstract PWE (which is obtained from an UPS) into an instantiated PWE. The trader will utilize the NIIS in order to get a list of available resources, from which it then chooses the ones best suited to build the requested PWE. That means that for example the abstract description “nearest printer” gets replaced by an actual printer resource. Figure 2 shows another example (in the context of the World Wide Web).

Some work has been done ([6], [8], [9]) concerning how traders might flexibly choose among a set of potential service providers (resources). One feature that is needed in this context is that the trader must be able to function with highly dynamic attributes. If the NIIS returns two accessible printers, but one of them is switched off, the trader should decide to have the other printer instantiate the abstract printer resource in the PWE description instead.

D. Globally Unique User-IDs

To be able to tell whether a user can actually access a given resource, a globally unique user-id must be presented to whatever security system guards that resource. It seems to us that an email-address might serve as such an id. In the address `user@host.domain.net`, host.domain.net (which today is the mail handling host) could also specify the user’s UPS host. That host can then be contacted to help authenticating the service request (e.g. the UPS server might be asked for a password, which is then compared with the password the user gave when accessing a certain service).

E. Mobile Intelligent Network Terminal

All network resources (services, hardware etc.) are found using the NIIS and made available by the PWE trader. Therefore, if applications running on the terminal access the resources they need through the PWE trader they will be able to act in the same (or at least a similar) manner regardless where the user is and what terminal he/she uses. In particular, the user environment will always be the same (subject to availability of needed resources, of course). A terminal that deploys the architecture presented in this paper shall be called Mobile Intelligent Network Terminal (MINT) because it “intelligently” provides the user with the same environment regardless where he/she is.

In the current system, we used Netscape as a first approximation for a MINT. See chapter IV to learn why we chose Netscape despite it being still far away from meeting all requirements of a MINT.

III. How these Components Interact

To demonstrate how these components interact we will make use of a well known example: a travelling salesman, except that this time the travelling salesman is not concerned with the distance travelled but with the global telecommunication network and with the problems that arise when moving

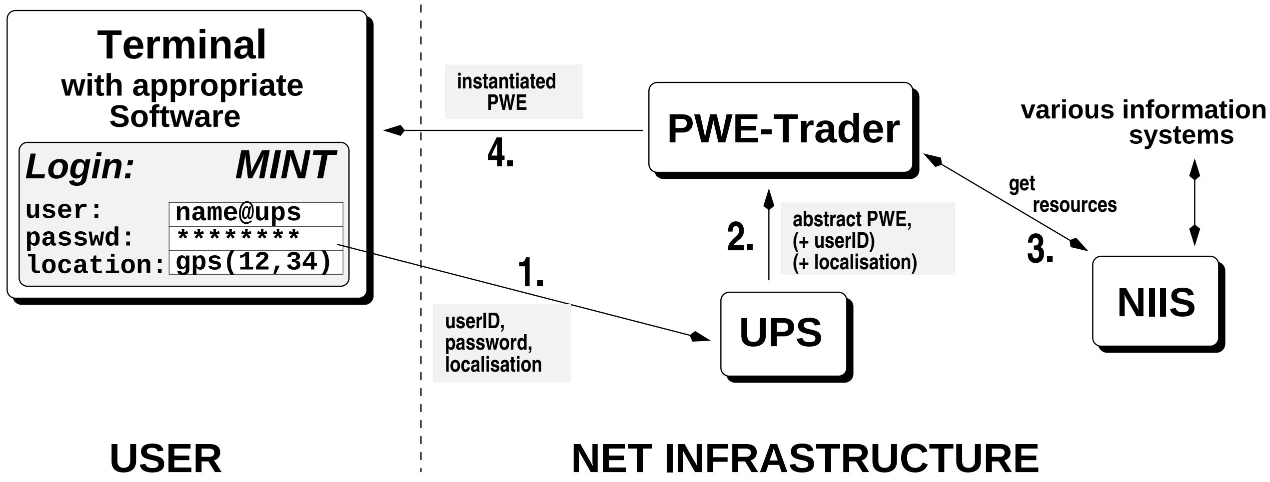

Fig. 3. How the components interact

from one point in the network to another.

Let us assume that our salesman is equipped with a laptop and visits customers and partners around the globe. In order to be able to gain access to the network his laptop features a cellular modem. Also, a global positioning system (in handy PCMCIA size) is built into his laptop.

Let us further assume that our salesman usually has meetings with his customers during which he needs to access the central database of his company. Apart from that, he ought to be connected to his email and USENET-news feed in order not to miss important information. Also, he might want to be informed about local cultural events, movie theatres, restaurants etc. for the (rare) evenings he spends out. According to these needs he will prepare his abstract PWE which, among other things, will contain static links to his company's central database and his email account. Abstract descriptions of potential resources to be taken from his respective location (e.g. a list of local movie theaters, available printers etc.) are also part of his abstract PWE.

Once with a customer, the first thing to do is log into the network. Any terminal (that has the necessary software) could be used for this. If the customer does not have a terminal, the salesman can use his own laptop. He uses the terminal to connect to his UPS server and authenticates himself, using his global user id[1] and a secure authentication mechanism. Additionally, the terminal might send information about its location (using its network address and/or the global positioning system). The UPS will then deliver the abstract PWE of the salesman to his PWE-Trader, which will employ the Network Infrastructure Information Service to get actual instances of abstract services described in the abstract PWE. In our case, this might concern a service listing the showtimes of local movie theaters. Other resources, such as his email account or the network address of his company's central database do not have to be instantiated by the NIIS, as they are independent of the salesman's actual position in the network and can be "hardcoded" into the abstract PWE. Obviously, the NIIS will need the supplied information about where the salesman is in order to find appropriate resources.

Finally, the PWE-Trader will pick resources from the list provided by the NIIS, turning the abstract PWE into an instantiated PWE. In our example this could mean that a link to a local movie theater schedule is added to the salesman's PWE and that a connection to the nearest printer is established. Hardcoded resources are copied as they are from the abstract PWE into the instantiated PWE (in our example, his email account and the database's network address).

Now, the terminal has to provide an appropriate user interface to the provided resources (or to signal that some resources could not be found). In our example, it could (temporarily) add the link to the local movie schedules to the salesman's Netscape bookmarks, set his PRINTER environment variable appropriately and start his email and database client applications.

Figure 3 picture summarizes the interaction of all the components.

IV. Experimental Prototype Implementation

In the current implementation, we chose Netscape[2] and the World Wide Web as the basis for a first prototype. There are a few reasons for this:

1. If it's not the Web, it can't be cool. More seriously,
2. Netscape is very common and at the same time still gaining momentum. There is basically no alternative to the Web if user awareness is an issue. As we consider the problems addressed in this paper to be seri-

1. As suggested, this may consist of the network address of the UPS server plus his id on that server.

2. Other browsers such as Mosaic, Lynx, Omniweb etc. can be used as well.

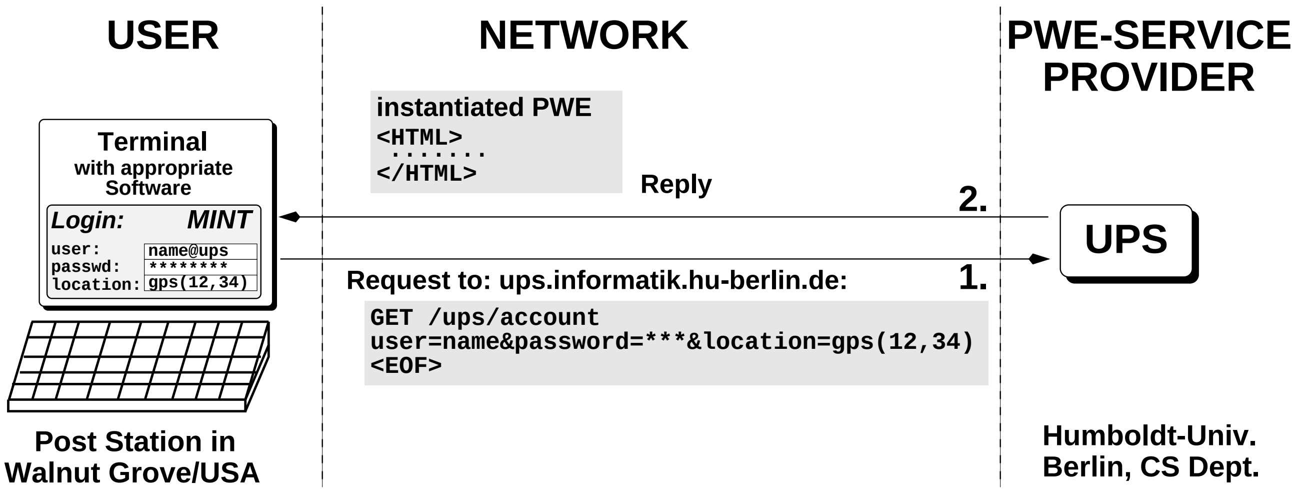

Fig. 4. Netscape as a Mobile Intelligent Network Terminal

ous ones, we of course want people to *use* the system.

3. Netscape (or any other browser, for that matter) provides a unified resource access: URL's. We use this property in our prototypical implementation.
4. Netscape is on its way to become the operating system of the Internet. It is still far away from that, but since its announced support for Java-applets, it is beginning to resemble features of an O/S: allocation of local resources, scheduling, protection etc. It is not clear whether Netscape will eventually be fit enough to be a full blown MINT, but at this time it seems to be a reasonable bet.

With Java-applets in place, Netscape could become the hosting "O/S" for the kinds of applications that we are talking about in this paper. However, the current releases of Netscape (lacking a better security policy) allow only very restricted use of applets (see, for example, [10]). Having globally unique user-ids and an infrastructure above them, our system might help to define a more concise security policy for Netscape and the Web. This could lead to a relaxation of the built-in strict security policy which in turn would move Netscape closer to what we envision as a Mobile Intelligent Network Terminal.

A. Terminal

As explained above, the network terminal that we use so far is Netscape. When a user wants to access his/her PWE (which is basically a Web page, similar to a "home page"), he/she has to connect to a log-in page of his/her UPS server, which in our case has to be a HTTP server. This log-in page (at, say, `http://ups.domain.net`) requests the id, a password and optionally location information from the user. This might be one example of such a page:

```
<FORM METHOD=POST ACTION="http://
ups.informatik.hu-berlin.de/ups/access">
User: <INPUT TYPE=text SIZE=33 NAME=user>
<BR>
Password: <INPUT TYPE=password SIZE=33
NAME=password> <BR>
Location: <INPUT TYPE=text SIZE=33
NAME=location> <BR>
<INPUT TYPE=submit VALUE="Enter network">
</FORM>
```

After posting this form, the UPS will engage the PWE-Trader (and in turn the NIIS) to provide the instantiated PWE which is an HTML document. Displaying this document, Netscape establishes the personal working environment that the user called for. Some of the hyperlinks will have been inserted by the PWE trader when it instantiated the abstract PWE. HTML hyperlinks can refer to all kinds of resources, including Java-applets, so that there exist various possibilities to represent the requested resources. Figure 4 shows an example.

1) Technical Details

Netscape can be configured to display the log-in page at start-up. This might even be a page on the terminal's local file system and therefore does not need to cause any network traffic. In chapter III.B we explained that the NIIS needs information about the location of the terminal. The protocol between terminal (Netscape) and UPS (httpd) in our case is HTTP which is based on TCP/IP. Therefore, the UPS server can acquire information about the network address of the terminal. This alone might not be sufficient, though, for example in the case of a user dialing into the network with his cellular modem. For that case our log-in page has a location field, which for example could take coordinates supplied by a global positioning system (GPS) in the following form:

`gps(x,y).`

To properly transmit the terminal's network address, Netscape must not use any proxy servers. However, if the proxy server runs on the same machine as the terminal (Netscape), or in the very near vicinity, the NIIS can still correctly report all available resources. This might be a good idea to minimize network traffic.

2) Future Directions

Our future work will definitely have to establish whether or not Netscape is flexible enough to seriously serve as a network terminal in the sense outlined at the beginning of this paper. Java-applets seem to be a promising step in the right direction, but still some work has to be done to turn Netscape into a powerful and general platform for networked applications.

The flexibility we are aiming at should eventually make it possible for example to have an audio version of the terminal (Netscape) which can be accessed by any telephone. Texts could be read out to the user, and links to follow could be chosen through a voice menu. This would allow blind or otherwise handicapped people to make use of the provided facilities.

B. UPS

As has been mentioned above, the User Profile Service has been realized in form of an HTTP-server. The log-in form is being POSTed and processed by a CGI script similar to this:

```
read INFO
USER=`echo $INFO | gawk 'BEGIN
{RS="&"; FS="="} /user/ {print $2}'`
PSWD=`echo $INFO | gawk 'BEGIN {RS="&";
FS="="} /password/ {print $2}'`
LOCA=`echo $INFO | gawk 'BEGIN {RS="&";
FS="="} /location/ {print $2}'`
LOCA="inet($REMOTE_HOST):$LOCA"          #
add inet access-point of terminal

# check $USER and $PSWD here!

if [ -r /u/ups/$USER/abtractPWE.html ]
then
    if [ -x /u/ups/$USER/pwe-trader ]
    then
      PWETRADER=/u/ups/$USER/pwe-trader
    else
      PWETRADER=/u/upsbin/std-pwe-trader
    fi
    cat /u/ups/$USER/abtractPWE.html |
$PWETRADER -u $USER -l $LOCA
else
    echo $USER not registered here
fi
```

After authentication, the appropriate abstract PWE is fetched (which, in our current implementation, is an HTML document on the UPS server's file system) and passed to the user's (or standard) PWE-Trader. Once the PWE-Trader has processed the abstract PWE (and turned it into a instantiated PWE), the instantiated PWE, which again is an HTML document, is returned to Netscape.

As we can see, the UPS architecture is inherently decentralized and we expect it to scale well in the context of global telecommunication networks. For testing purposes we have provided a UPS server at Humboldt University (`ups.informatik.hu-berlin.de`) and a guest account (User: "guest", Password: none). The globally unique user-id of this user is therefore `guest@ups.informatik.hu-berlin.de`.

1) Future Directions

The functionality provided by the UPS so far is sufficient for testing the concepts presented in this paper. However, one notable feature of the current UPS is that is doesn't preserve state across multiple sessions. In the future, the UPS might be a place where the user not only stores a description of his/her working environment, but also his/her files, electronic cash etc. Accessibility to these things would then become independent of the user's terminal actually being connected to the network.

Also, the UPS might, depending on the network connection (high vs. low bandwidth) or the terminal used (laptop vs. telephone), employ different PWE-Traders which are specialized to look for resources suitable for the given situation.

C. NIIS

The Network Infrastructure Information Service has not been implemented yet in the form of capter III.B. Instead, we use a simple database of resources that can be queried by location information. So far, this database is centralized and contains information about all participating sites. We have implemented location-resource mappings for IP-addresses, GPS-coordinates and telephone numbers. For each of these there exist separate databases which correspond to the specialized location services of chapter III.B.

D. PWE-Trader

We recall that the abstract PWE in our prototype implementation is an HTML document. This document, using well-known HTML tags, contains links to some statically defined resources. Resources that are to be instantiated by the PWE-Trader are marked by a new HTML tag: `<pwe>` *description* `</pwe>`.

In between the <pwe> and </pwe> tags the user can describe what sort of resource should be substituted at that point. The possible forms that *description* can take are, rather informally, described as follows:

```
description:   resource ("|" resource)*
resource:      resourceName (attribute)*
attribute:     attributeName op value
op:            "="|"<"|">"|"<="|">="|"<>"
value:         string | integer | real
```

Let us assume that we wanted to specify a printer. Because we want high quality print-outs, we are requesting a laser printer with at least 300dpi. If that is not available, we would consent with an ink jet printer, provided that it is in color and that the price per page is less than 10 cents. The description of these requirements in an abstract PWE look like this:

```
  <pwe>
printer type=printer dpi>=300 |
printer type=ink-color price<10
</pwe>
```

The PWE-Trader passes a list of required resources to the NIIS. The NIIS returns a list of potential providers but does not yet decide between them. Then, the PWE-Trader selects the providers that are best suited and inserts their representation into the PWE. These representations (i.e. HTML constructs) are chosen so that, upon reception of the instantiated PWE, Netscape displays those resources in an appropriate manner.

1) Future Directions

At the moment, the possibilities to describe resources are severely limited. We are working on a much more flexible way to describe requested resources. For example, we want to be able to specify that we want both a color terminal **and** a color printer, and if we can only get a black-and-white printer that we then want a black-and-white terminal as well.

Also, the user should be able to specify how the requested resource is being represented. So far, the PWE-Trader makes this decision. Finally, the PWE-Trader might want to choose representations depending on certain circumstances. For an expensive, slow link through a cellular modem, off-line email clients might be more suitable than for a cheap, fast Ethernet connection.

The current PWE-Trader component can easily be replaced by a more advanced design. Apart from that, each user has the possibility to provide his/her own PWE-Trader. This makes our implementation open to the enhancements suggested in this chapter.

V. Experiences

We have been using a UPS server at Humboldt University for a small period of time now. Because the UPS server basically consists of an (already existing) HTTP server, the prototype was set up rather quickly. The big advantage of the implementation as it is now is that it uses Netscape as the terminal and all custom-made software (UPS, NIIS, PWE-Trader) exists on the server only. So, everyone who has Netscape can use our system *now*. Furthermore, users are already familiar with Netscape, so they do not have to learn a new system.

So far, abstract PWE's are maintained by an administrator. This proved to be not very convenient. In the near future we will have to provide means to the users that will allow them to create their own abstract PWE's and store them with their UPS.

A PWE tends to limit the user's view of the network (that even was one of the goals). If interesting information becomes available that a user has already given up looking for, sticking to a PWE might prevent the user from noticing the appearance of that information. We therefore suggest that all PWE's include links to Internet search engines such as Yahoo or Lycos, in order to combine the best of both worlds: Access to the whole Internet and a personalized, structured view of it.

The biggest problem turned out to be the Network Infrastructure Information service. Even to cover an only moderately big institution such as a Department of Computer Science a large amount of information had to be collected. If NIIS information is missing for a certain location, the worst that can happen is that the PWE-Trader does not automatically provide requested resources. The user is then back where we all are today: Facing a huge, chaotic Web full of information that we are not interested in.

VI. Future Work

In chapter IV we already pointed out future directions for the components of our architecture. We especially hope to be able to employ Java-applets in a less restricted way than is possible today in order to make Netscape a true Mobile Intelligent Network Terminal.

We will also work on support for mobile terminals (laptops) connected to the Internet through various wireless networks. Connections through a cellular network tend to be more expensive and provide less bandwidth than usual connections. We will have to find service representations that account for that fact. [5] for example describes how proxy servers and "intelligent" prefetching can be used to allow off-line work. As a basis for our research on wireless communications we have chosen IBM's network platform "ARTour" [7] which provides robust TCP/IP connectivity across different cellular networks[1].

VII. Conclusion

The abundance of information currently available on the Internet (and especially the World Wide Web) calls for means to bring order into this apparently chaotic system. In this paper we have introduced an architecture that allows each user to establish his/her own notion of a "non-chaotic Internet". The proposed architecture is flexible enough to account for changes of the user (i.e. especially changes in their location) and changes of the network (e.g. development of the network infrastructure).

For local networks, there exist solutions for many of the emerging problems. However, these solutions are usually dependent on a centralized administration and do not scale well in a global context. Our architecture is simple, non-centralized and open to various enhancements which we believe makes it a promising approach for global telecommunication

1. Networks supported so far include Modacom, GSM and Mobitex.

networks.

The utility of our architecture has been tested in a prototypical implementation at Humboldt University.

References

[5] A. Schill, S. Kuemmel. *Design and Implementation of a Support Platform for Distributed Mobile Computing*. Distributed Systems Engineering Journal, 1995.

[6] R. Popescu-Zeletin, V. Tschammer, M. Tschichholz. *'Y' distributed application platform*. Computer Communication (14), p. 366 ff., 1991.

[7] IBM Deutschland Informationssysteme GmbH, WT Mobile Data Communication - ARTe. *IBM ARTour, Technical Overview, Release 1*. 1995.

[8] J.-P. Redlich, D. Balfanz. *Concepts for a Flexible Service Architecture*. Proc. IS&N'94, Aachen, LNCS Vol.851. Springer, 1994.

[9] ISO/IEC JTC1/SC21/WG7. *Working Document on Topic 9.1 - ODP-Trader*. July 1993.

[10] D. Wallach, D. Dean. *Security Flaws in the HotJava Web Browser*. Princeton Computer Science Technical Report 501-95. 1995.