VirtualHome

Göran Hultgren and Jonas Öhrn

Bluefish AB

28 March, 2002

Abstract

VirtualHome is about introducing the concept of *personal accounts on the Internet*.

A VirtualHome account stores *resources* which can be of almost any kind - files, emails, documents, bookmarks, addresses, calendar information etc. It can also hold agents and applications that have *active behaviour on behalf of the account owner* and can perform assignments when the owner is not personally online. The accounts are hosted in a network of VirtualHome servers but can easily be moved from one server to another.

The servers, accounts and resources all share a common address space based on 'tumblers' which are identifiers *guaranteed worldwide unique* and *infinitely extensible.* The use of tumblers in combination with restrictive rules for tumbler reallocation and address forwarding ensure that *servers, accounts and resources are guaranteed resolvable* avoiding the current problem of broken links on the Internet.

The owner can access the account using a variety of protocols, but it should be as transparent as possible to the user that the resources are not stored locally. One mechanism to achieve this transparency is *replication and synchronization of resources* over multiple client machines.

The goals and intentions of the VirtualHome project are:

(1) To create a set of well defined VirtualHome protocols and specifications.

(2) Creating a working reference implementation of the VirtualHome components.

(3) To explore active behaviours suitable for personal accounts.

(4) To investigate alternative user interface metaphors for VirtualHome accounts.

VirtualHome

Göran Hultgren and Jonas Öhrn

Bluefish AB

28 March, 2002

1. Introduction

VirtualHome[1] is a new open infrastructure to handle 'personal accounts on the Internet' that holds an individuals all personal digital information.

1.1. The problem

Before the personal computer took hold of the desktop the concept of a 'personal account' was used to group information belonging to different individuals, even though they where physically stored on a single computer. Anyone accustomed with multiuser systems like for example UNIX recognizes this concept. These accounts hold all information that belongs to a particular user on that machine or a whole network - documents, email, personal settings etc.

The personal computer changed all this when the common scenario became that each individual had a computer of their own. Users got accustomed to storing documents locally on their own computer - the computer turned into a very personal 'possession', in essence it overtook the role of the personal account.

This phenomenon should not be underestimated - the feeling of ownership and full control is something that users of personal computers do not want to give up. Any new technology will have to take this into account.

The current focus on personal computers creates problems when the user has to use more than *one* computer. People working professionally with computers already likely use much more than one computer and as the Internet becomes even more accessible through the use of PDAs, laptops, mobile phones etc the problem will become even more widespread.

In short - each individual tends to build a 'personal space' of digital information; email, documents, notes, funny little programs, address books, calendars and so on. Ideally we would want to have all this information accessible wherever we are and however we are connected to the Internet.

1.2. The solution

VH is about reintroducing the concept of a personal account, but this time on a global scale, on the Internet. VH is focused on the needs of the individual, not on the needs for large web sites, central document stores, group collaboration, project management etc.

The concrete goal is to establish *a set of protocols and specifications* together with *a func-*

[1]The acronym *VH* is used throughout the rest of this document to denote VirtualHome

tional, free, OpenSource reference implementation that hopefully will create enough momentum that it will get a life of its own and evolve further in an OpenSource setting.

By looking at existing successful technologies of the Internet we try to create a new infrastructure of protocols and specifications that are capable of sustaining our vision of VH - *mobile personal accounts transparently accessible everywhere* - yet being simple and cross platform enough to be *easy to understand and implement*. There are too many over complicated failed standards out there that proves the simple rule *simplicity is key to success.*

2. History

This chapter presents how the ideas around VH has evolved at Bluefish during the last 3 years up to today. The project has been 'hibernating' from time to time due to lack of time, intervening assignments etc but the same people have been involved from day one.

2.1. The seed

The ideas about VH started about 3 years ago with the explosion of interest in mobile computing. The initial seed was the fact that all these small devices that people soon would have connected to the Internet just didn't have enough bandwidth to be able to sustain 'normal' Internet activities.

We started sketching on a system where there was a server placed on the Internet acting as a 'robot' on behalf of the mobile device doing the actual Internet communication and then communicating the results in a highly optimized and compressed way to the mobile device. On the device we envisioned having what we dubbed a *universal proxy* that would be able to translate the optimized protocol into standard Internet protocols/formats like SMTP/POP3/HTTP/HTML etc. enabling the user to be able to use ordinary Internet tools on the device that assumes less limited bandwidth.

Essentially we wanted to apply the positive effects of a good client/server architecture[1] in the world of limited mobile clients.

2.2. The plan

Employees at Bluefish are involved in a number of OpenSource projects both as users and developers and we are fully aware of all the advantages that an OpenSource setting can bring to a project. Early on we decided that if the ideas around VH would stand any chance to become something larger it would have to be developed in a community under a *free license*[2].

Still we wanted to move our ideas and vision of VH into something more tangible like a working prototype before we took the step of trying to create a community around the ideas and the implementation of the vision. Experience shows that a successful OpenSource project needs at least a somewhat working base implementation before developers get interested enough to join the project in any large number.

---

[1] Various companies have since then implemented variants on this theme for example to enable web surfing from a device running Palm OS with both limited bandwidth and display capabilities. The WAP gateways of today are also clearly something along these lines.

[2] The definition of the word 'free' is open to much debate in the OpenSource community. In this context we refer to licenses of the BSD/MIT/X11 style, not viral licenses like GPL.

Beside developing the reference implementation of VH under a free license we also wanted the project to have a strong focus on *protocols and specifications in combination with test suites that can verify them.* A protocol or a specification, however simple and clearly documented, can never be assumed to produce compliant implementations and if a flourishing market of implementations is expected to be possible we need complete test suites for all components that can verify compliance to the definitions.

Finally it's an art to nurse an OpenSource community so that developers stay focused and interested in the project, new developers are attracted and can contribute in a reasonable amount of time and so on. This should not be underestimated.

2.3. The personal account

After the first few weeks of brainstorming around VH it became apparent that a more profound focus of VH was the idea of having your personal digital information at hand wherever you are and however you are connected to the Internet.

After our shift in focus we investigated quite a few of the available services on the Internet today and came to the conclusion that even if some services exist like web mail services or so called netdrives, all of these services are proprietary and all of them focus on only one aspect of personal information.

2.4. Network of servers

We had always envisioned a server software sitting on the Internet acting on behalf of the user that possibly would be connected to the Internet with much less available bandwidth. We had not thought that much about how these servers should interact with each others or what other responsibilities they should have. We did however view them as hosts for the personal accounts which would be active even when the user was not connected.

We started thinking about mobility of personal accounts and the idea that a user should not be locked to a particular server provider. The personal account should be easily moved to another server or provider if the user would want to. Since future hosting companies typically would not want their users to change to another host, the mechanism for moving an account would have to be carefully specified and verified by the VH protocols and test suites. A host company would not be able to stamp their service 'VH compliant' unless their servers supported the test suites to the full extent.

There are two big problems with the network of HTTP servers on the Internet today:

1. There is no way to enumerate the servers available on the Internet[1].

2. References[2] to documents on HTTP servers are based on location so a document can not be moved without invalidating the reference.

The first problem means that there is no way to produce a catalog of all HTTP servers. The

[1] The Domain Name System on the other hand is enumerable - we can produce a complete listing of registered domain names.

[2] Uniform Resource Locators, URLs.

second problem also means that the referee can not tell if a document has been moved or removed altogether. This is actually also true for the servers themselves.

2.5. Xanadu and tumblers

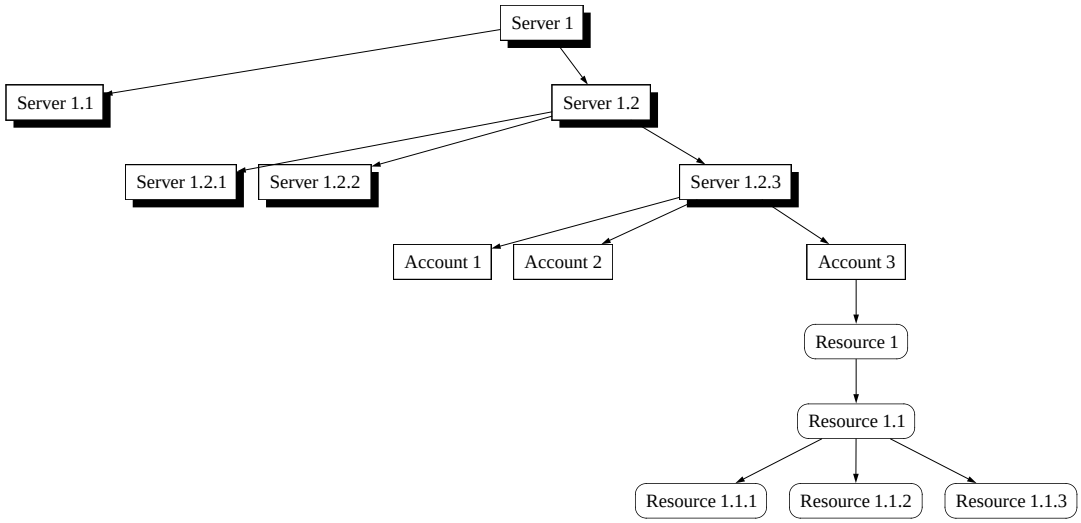
We started investigating alternatives to the current scheme with URLs and stumbled upon the Xanadu project. Xanadu has a rather different focus than VH but some of the fundamental aspects are the same. Xanadu had devised an addressing using tumblers. In short, a tumbler is a series of integers separated with dots, much like an IP number. But there is no limit to the integers maximum value nor the number of integers in the tumbler. A typical tumbler can look like this:

1.2.3.0.3.0.1.1.3

Here we chose deliberately rather small values for the integers but they can be arbitrarily large. The zeros acts as delimiters so this tumbler actually has three parts:

1.2.3.0.3.0.1.1.3

In VH (and similarly in Xanadu) the first part refers to a server, the second part refers to an account on that given server and the third and last part refers to a resource within the given account. This can be visualized as a strict tree with three different 'levels' like this:



The original idea in Xanadu was that once a document (as a resource in VH) was created it would get a tumbler and the document would never be deleted. In VH we are considering a set of rules that will ensure that a tumbler can always be *reasonably* resolved:

- A server tumbler can never be reused. A server can be removed but the list of accounts residing on a server should be redundantly stored on one or more other servers in the network. And if a server is removed a server redirector (placeholder) is left in the network.

- An account can move to another server but this leaves an account redirector in it's place on

the original server. This means that the original account tumbler can never be reused and future references through that tumbler will be rerouted to the new location of the account.

- Resources can be removed from an account but the tumbler of the resource can never be reused within the account.

The above rules (or some variety of them) will ensure that a reference to a resource within the VH network will always resolve in one of the following ways:

1. Resource (and thus server and account) fully resolved and found.
2. Server temporarily down - basic account information is available through a backup server but the real account and it's resources are unavailable.
3. Account found but resource removed. A note can be left behind but it is not necessary.
4. Account removed with a note left behind.
5. Server removed from network and account lost. (If an account is reconstructed from a removed server a redirector on the parent server will be able to reroute to the new account)

We can never protect from the fact that a server can go down, but the system can be made somewhat redundant by forcing a server to synchronize information about the available accounts to at least one other server.

A resource should also be removable but since we never reuse a tumbler within a specific account (and we never reuse account tumblers nor server tumblers) we can rest assured that it really has been removed if it isn't found. Moving resources (and servers or accounts) are done by placing redirectors behind.

2.6. Resource access and sharing

Below are some of the many ideas we have on different types of resource references possible in VH:

- Redirector
- Reference
- Link
- Mirror link

A redirector is a placeholder for a resource that is inaccessible for a reason, most commonly because the resorce have been moved or deleted. A redirector can advertise the new location of a resource or a note why it has been removed.

A reference refer to another resource by its id/name in the same way Unix 'symlink' works. This is probably the most common way to reference resources owned by other users. The referenced resource can exist anywhere within the VH world.

The resource is deleted when the last link to it is removed. This assures that the resource remains in a users account when another user ‘deletes’ or unlinks the same resource in the user’s account. A link to a resource in another user’s account can be created only if that is allowed by the owner. A linked resource can be viewed of as a shared single copy of a resource (even though a resource in VH is always owned by one and only one user).

Mirror links holds a copy of a foreign resource for mirroring purposes. The foreign resource is copied according to a specfied schema, e.g. once every hour or when advertised as changed by the foreign server. You can view status and differences between the original and the mirror uses rsync-like algorithms to ensure minimal network data transfer to synchronize.

A resource can always be addressed using its global VH id, i.e. the tumbler. We have also planned to employ a default user defined naming of the resource simply by using the name of the resource in the context of the user account and resource location. A user can name a resource as ’mydoc.txt’ hence allow this to be addressed as ’server.0.account/mydoc.txt’ where ’server.0.account’ is the tumbler of the account.

2.7. Simple protocols

In our work with VH we are focused on constructing Internet protocols that are easy to implement and that will attract as many developers in as many languages and environments as possible. We are familiar with many of the modern higlevel communication frameworks like CORBA, RMI etc but all the successful cross platform Internet protocols are clear text Socket protocols.

One of the base principles in VH is that the account and the resources should be accessible through as many different protocols as possible. This doesn’t rule out future complementing implementations using mentioned technologies but the primary implementation should be based on a simpler level.

Advantages of using clear text socket protocols are:

- Good performance.
- Easy to debug since the traffic is not binary and can easily be inspected.
- Sockets are available on essentially every network capable programming language in existance.
- The knowledge burden on the developer is minimal and the focus is on the protocol and not the underlying communication technology.

2.8. Bookmarks and BTP

In order to have a sample resource type to experiment with we chose ‘Internet bookmarks’ since it’s something that you typically want to be able to access wherever you are and it is a rather simple resource type to implement.

We started by realizing that there is no standard protocol available today for a client application. Email applications have SMTP/POP3/IMAP, but there is no equivalent for a bookmark client application. In order to impelement a correct architecture we specified and implemented

a simple but efficient protocol called BTP (Bookmark Transfer Protocol) that uses incremental updating in order to keep a client synchronized with a server holding the bookmarks.

We have used this simple protocol as a workbench to explore different ways of specifying and implementing clear text socket protocols.

2.9. Today

The current status of VH is that we are *2-3 developers* on Bluefish that have spent about *800 hours*. The results sofar are:

1. A base implementation of tumblers, the account server, accounts, resources and more. It has a web UI to begin with for people to create/administer and delete accounts etc.
2. A handy and novel web application framework based on the freely available Squeak webserver Comanche. This framework enables us to prototype web user interfaces in a very efficient manner.
3. A specification and full implementation of the BTP protocol.
4. Three different reference implementations of BTP clients (2 in Squeak, one in C++).
5. A first embryo of an implementation of the ‘universal proxy’.
6. A range of design documents describing different ideas and areas within VH.

Total amount of code is about *10000 lines* for VH and about *2000 lines* for the web framework.