Fog Creek SOFTWARE

Project Aardvark Functional Specification

Last Updated: August 17, 2005

Table of Contents

Foreword to the Public Edition, by Joel Spolsky

As the final version of Copilot.com (what this spec calls "Aardvark") went into production in early August, 2005, this spec is now of historical interest only. I'm making it available as a part of *Joel on Software* because many people have asked to see some samples of the kinds of specs we write at Fog Creek.

There was one big mistake in the spec. When I wrote it I had assumed that port 443, used by the secure (SSL) http protocol, would be open even on networks that are configured to use proxy servers. I mistakenly believed that proxy servers could not forward SSL traffic because that would appear, to the browser, to be a "man in the middle" attack. This, it turns out, was erroneous, false, and incorrect, and after shipping the first beta we discovered just how wrong it was. Thus, the final implementation of Copilot 1.0 required a lot of unplanned work to support proxy detection and different kinds of proxy servers. We got most of that work done in time for 1.0 and are now finishing off the rest of it for a release we're calling 1.1.

That said, the extra proxy work only added about 10% to the overall development effort. The reason you write a spec is not to solve every possible problem in advance: the reason you write a spec is to solve as many problems as you possibly can in advance so that you minimize the number of surprises that come up during development. In our case, since we were working with summer interns who literally had to walk out the door and go back to school, we literally could not afford to underestimate the amount of work that was needed to build Project Aardvark or *I* would be stuck debugging the thing deep into September while the kids drank beer and played poker, and we couldn't have *that*.

Many times, thinking things out in advance saved us serious development headaches later on. When I wrote the first draft of this spec, I had a more complicated flowchart that assumed that either party (helper or victim) could initiate the process. This became extremely complicated and convoluted for various reasons. As I worked through the screens that would be needed to allow either party to initiate the process, I realized that Aardvark would be just as useful, and radically simpler, if the helper was required to start the whole process. Making this change in the spec took an hour or two. If we had made this change in code, it would have added weeks to the schedule. I can't tell you how strongly I believe in Big Design Up Front, which the proponents of Extreme Programming consider anathema. I have consistently saved time and made better products by using BDUF and I'm proud to use it, no matter what the XP fanatics claim. They're just wrong on this point and I can't be any clearer than that.

There was one other significant change we made late in the process which is not reflected in the spec. After usability testing, we discovered that people were consistently confused by our policy of having the application delete itself. This was not what users expect. We modified the final product so that the helper could check a box to cause the victim's executable to be deleted, but it wasn't done by default. This way the helper, who is presumably more computer savvy, could insure that no unsafe executables were left around on the victim's computer. I think this change was about 2 hours of work, so it didn't impact the schedule.

Finally, you'll notice a big "coding conventions" section in the middle of the spec. Frankly, this doesn't belong in a functional specification, and it should really have been a separate document.

Still, as I reread this spec today, I'm surprised by what a good job the Aardvarkians did implementing everything. I seriously expected we would have to cut features; instead, we shipped with everything we wanted in version 1 and even did some of the features this spec refers to as version 2 features.

-- Joel Spolsky, August 17, 2005

About this Specification

This specification is simply a starting point for the design of Aardvark 1.0, not a final blueprint. As we start to build the product, we'll discover a lot of things that won't work exactly as planned. We'll invent new features, we'll change things, we'll refine the wording, etc. We'll try to keep the spec up to date as things change. By no means should you consider this spec to be some kind of holy, cast-in-stone law.

Overview

"Project Aardvark" is the codename for a new product to be developed at Fog Creek during the summer of 2005. The actual name of the product will be different.

Aardvark allows people to help their friends, relatives, and customers with their computer problems by temporarily taking over their computers over the Internet.

For the purpose of internal documentation, we will refer to the person doing the helping as the *helper* and the person whose computer is being controlled as the *victim*.

Here's a brief summary of how Aardvark works:

1. The helper goes to the Aardvark website to register and make payment arrangements.

    - They can pay with PayPal, a credit card, they can get a quick trial for free, or they can request that the victim be asked to pay.

2. Aardvark gives the helper an *invitation number*. They read this over the phone to the victim.

3. The helper downloads and runs a small piece of software.

4. The victim goes to the Aardvark website and enters the invitation number. They then download and run a small piece of software.

5. The software asks the victim for permission to have their computer controlled.

6. The software allows the helper to control the victim's computer.

7. When they are done, either party can disconnect. The software deletes itself when you exit.

Major Features

- Easy to get started:
  - No software needs to be permanently installed
  - Simple fast payment and no commitment
- Version 1.0 is Windows only and requires a reasonably broadband connection
- Blasts through all firewalls as long as outbound connections on port 443 are allowed. This is the port used for secure websites (https), and virtually every firewall allows traffic through port 443.
- Secure (hopefully).

Design Goals

All design and engineering decisions will be taken with the following principle in mind:

Simple is better than complicated.

We will always try to eliminate choices that the user must make. Every bump on the user experience must be relentlessly sanded down. Our customers will say: "It's the easiest thing in the world and It Just Works." No jargon will remain in the user experience, anywhere.

Major Components

The Aardvark system runs on servers which are hosted by Fog Creek Software. There are four major software components in Aardvark.

The Website

A database-backed website used to register with Aardvark, make payment arrangements, and set up remote control sessions.

Victim Software

Software run by the victim. This is a small, self-contained Windows EXE that the victim downloads and runs. The code for the victim is based on VNC which is GPL, so this component will be re-released under GPL.

Helper Software

Software run by the helper. This will probably be a small, self-contained Windows EXE that the helper downloads and runs, but in v2 we may also make available a Java Applet that the helper runs through a web page, or an ActiveX control, or a Firefox extension: whatever makes the end-user experience the most seamless. The code for the helper is also based on VNC.

The Reflector

A Windows Service which we run on our servers, used to allow any helper to help any victim even when both of them are behind firewalls. Both helper and victim connect to the reflector. The reflector checks that they are authorized and relays messages between helper and victim until the paid-up time runs out.

Licenses

VNC is GPL. The two components we're building based on VNC, the helper and victim, will need to be re-released under GPL.

This is not that big a deal. The code will be highly optimized for our own use and will *require* the reflector to work, which will *not* be released under GPL.

About The Rollout

Our primary goal is to rollout a fully-functional version of Aardvark in only one summer. That means that the initial version will almost certainly not be optimal in every way and will not contain every feature. Throughout this spec, the notation **(v2)** will be used to indicate features which will not be a part of the initial product but which will be added later.

Physical Architecture

The initial rollout will go onto our existing web servers -- high end, well-endowed Dell 2850s – each of which can handle the entire Aardvark load by itself if necessary. The system will be highly scalable so we can add additional servers progressively to handle more load. Our web servers are named web1 and web2. We will move rapidly to add servers if the service is popular.

Scalability, load distribution, and some failover capabilities are provided by Windows NLB (Network Load Balancer). NLB is about as simple as you can imagine: web1 and web2 both have their own IP addresses, but they also both share a third IP address, webnlb.fogcreek.com, and they coordinate amongst themselves to decide who will handle each socket connection that comes in on that shared IP address. The current system causes 50% of the new connections to go to each server. If one server is taken offline, 100% of the connections go to the other server. We can scale to about 16 servers comfortably. If we need to scale to more than 16, we have to start building blocks of 16 servers, and use a hardware load balancer to balance between each block of 16.

To prevent state bugs, every user that comes in from the same class C range of addresses will be served by the same server as long as it is available. This is a feature of NLB which we've already turned on.

The servers will be running Windows Server 2003 with SP1, IIS6, and ASP.NET. For a database, the servers will use our existing SQL Server machine which they share with everything else.

Each server will be running both the website and the reflector.

Because it will often be necessary to upgrade the servers, the system is designed so that taking down any server will have no effect and the remaining server(s) can handle the load. In order to take down a server, we need to drain it, allowing all current connections using that server to finish gracefully, while all *new* connections go to some different server. Once a server is completely drained it can be upgraded and then brought back online. Draining is built into NLB so no additional code is necessary to make that work.

(v2) Because remote-control sessions may last a long time, it will take quite a while to fully drain a server: possibly on the order of hours or even days. I think the best way to deal with this is to add code to helper/victim software so that if the connection ever goes down to the reflector, it automatically attempts to reconnect. That way when you need to upgrade a server, you can drain it of all the web sessions (which should only take a minute or two), and then just abruptly drop all of the reflector sessions and reboot. The helper and victim will both notice that the connection dropped and automatically reconnect to each other through a different machine.

Software Architecture

The website is an ASP.NET application with code written in C#. We will use Visual Studio .Net 2003 with version 1.1 of the .Net Framework.

Web Pages: Web pages will be standards-compliant to the extent allowed by ASP.NET. We will use CSS for formatting but *not* positioning since CSS positioning is too buggy on modern browsers: we'll just use tables for most positioning. Strings will be isolated from day 1 for easily localization.

The helper and victim are based on VNC and thus written in C++. We will try to use the Visual Studio 2003 version of C++ to compile it.

(v2) The initial version of the helper and victim are Windows EXEs only. In the future, we may want to also add:

1. A Java applet version of the helper
2. ActiveX versions of the helper and victim, depending on how the latest version of Internet Explorer deals with these. The only question is whether it's more seamless to give someone an EXE to run or give them an ActiveX control.
3. Helpers and Victims for Mac and Unix.

The reflector will be built as a Windows Service in C# so that it can share as much code as possible with the website, for example code for connecting to the database and handling user authentication and registration.

Coding Conventions

Since the website will be load-balanced, ASP.Net session state should be stored in SQL Server, not in memory (the default).

Always use the same name for the same thing. For example, the name of an INPUT in HTML should correspond to the name of the variable it gets stored in, which should correspond to the name of the column in the SQL database.

In C# code use Apps Hungarian, a.k.a. Aardvark Ukrainian. Having a stronger coding convention for names helps keep code correct, but it also helps you remember variable names because there's less variability, and it saves mental stress when you have to figure out a new variable name because so many are predictable.

Variable names are lowerCamelCase, e.g. prefixName

Common prefixes:

s = a Unicode string
ch = a single Unicode char
a = ANSI modifier. as = ANSI string; ach = ANSI char.
utf8 = A UTF8 string
c = count of something. So cb = count of bytes (buffer size), cch = string length
ix = index – the unique primary key of a database *or* an index in a loop *or* an index into an array
rg = ("range") – an array, usually as a prefix. For example rgs is an array of strings
f = Boolean flag, always true/false. Although traditional Apps Hungarian allows three-state flags we don't.

fl = File
fn = Function (when you have a function pointer)
path = Path, the full path name of a file
n = *n*-way flag or enum
b = byte.
db = database
dbl = floating point number
l = "long", by which we mean, a long integer that is not a count of something
x, y = x or y coordinates in pixels.
d = difference. Commonly used in dx or dy to indicate width or height, but could also be used as ddt (timespan), dix (offset in an array)
dt = date/time (includes both)
rs = recordset
sql = sql statement
tbl = database table
url = a complete URL.
urls = a string which has been URL-encoded
html = a string which contains HTML and is suitable to dump to the browser as-is
p = pointer modifier. Used for a pointer in C/C++; not needed in C# code.
o = object.

When you define new classes, pick a Ukrainian prefix to use for that class of 2-4 letters, and document it at the top of the class definition. So for example if you defined a class called CSession you might decide to use "sess" as the prefix.

Common names or suffixes:

Min, Max = minimum and maximum
Prev, Next, Curr = previous, next, and current
Tmp = a temporary value
Src, Dest = when copying things

The name part is often omitted when there is only one local object of the type you're dealing with or one which is most important. For example, if you write a string function that takes one string argument, it's just going to be named s. The arguments to a function that writes a string to a file would obviously be named (fl, s).

Function names are UpperCamelCase. Use TypeFromType for conversion functions, for example, Utf8FromS would be a function that converted Unicode to UTF-8.

Class names start with C or E. (*Objects o*f that class should use their own prefix which you coin.) Use E for entity classes, that is, classes which correspond to a single row from a table or view in the database. Use C for all other classes.

SQL Tables are always UpperCamelCase and named in the singular, for example, Customer and Person, never Customers and People.

Virtually every table contains an autonumber (identity) it is always the leftmost column, is always the primary key, and always named ixFoo where Foo is the table name. For example if your table is called Customer there is certain to be a PK named ixCustomer as the first column. If any other tables have foreign keys pointing into your table, they will be named ixCustomer, too, unless they have several, in which case, append a modifier. For example ixCustomerFirst, ixCustomerCurrent, ixCustomerReferrer.

In SQL tables, we use the same prefixes as we use in code. The most common are dt, f, n, c, and ix. Strings are usually just s unless html or url is more appropriate, but we sometimes use txt for large text (memo) fields. All SQL tables start with tbl, views start with view, and we hope never to use stored procedures.

The User Experience

We can assume that the helper and victim are already talking to each other somehow, probably by phone, IM, or Skype.

Once the helper decides to help the victim, he goes to our website. For the purpose of this spec we'll pretend that the domain name is **aardvark.com**. Although the domain aardvark.com is *very* underutilized, and can probably be bought, that probably won't be the final domain name. Our goal will be to find a name that is very easy to read over the phone, and unlikely to be misheard, mistyped or misspelled by the victim, and we'll try to register the most obvious typos.

So the helper goes to **aardvark.com**. That page has three things prominently: an explanation of what Aardvark is and what it does, a place for the helper or victim to initiate a help session, and a place to enter an invitation code.

The helper goes to initiate a help session. They are asked how they want to pay. The details of payment amounts and payment options still needs to be worked out. For now we can assume that some subset of the following options are available:

- Short remote control session (maybe about 3 minutes, limited to 3 sessions per 24 hours per IP address). The idea is that this too short to do anything complicated, but useful for seeing how the service works. Requires absolutely no signup or forms or anything. Just click the link and get an invitation code. Will definitely be implemented in v1.

- Single remote control session. Maybe limited to *n* hours. Can be paid for with PayPal or a credit card. PayPal, I think, will have the advantage that you don't need to reenter personal information. Probably around $5 or $10.

- Multi-packs of remote control sessions. Maybe 10, 100, and 1000 packs.

- *n*-hour pass, which you can use to help as many people as you want for *n* hours. Maybe available as 3 hour, 24 hour, 1 month, 1 year.

- Subscription service for people who provide lots of help. $x per month per concurrent user. You can make as many accounts as you want (with email/passwords) but the most concurrent sessions you can have at a given time will be limited by your subscription.

For simplicity sake we almost certainly won't offer all these combinations at launch time.

Note that either the helper *or* the victim can pay. Although in typical tech-support scenarios you would expect the helper to pay, for consumer usage it almost makes sense for the victim to pay, since after all, they are the one getting helped!

The customer is now taken to a page that tells them their invitation code and instructs them to instruct their friend to go to aardvark.com and enter that invitation code.

Both parties are taken to pages to download and run the software. That download gives them a custom EXE that has been modified to contain all the parameters it needs to connect back to the reflector.

There are two versions of the software: a version for the helper and a version for the victim. The helper software is really a VNC client. The victim software is really a VNC server.

The victim software, when run, asks the user for permission for the helper to connect to their computer.

Both EXE's connect to the reflector. Until both parties are connected, they display a message saying, approximately, "Waiting for your party to connect." They also provide instructions for what the *other* person needs to do, in case the other person gets screwed up.

When the victim runs their application, they are asked if they want to allow the helper (mentioned by name and optionally organization) to control their computer. They have to click "Yes" or the program will exit and self-destruct.

If the victim manages to connect first, they see a message that says "Waiting for helper to connect."

Once both parties are connected the helper application changes to show a copy of the victim's screen. The victim application appears as a small window with a single button that is used to close the connection.

When either party terminates the connection, the victim application self-destructs, that is, it deletes itself.

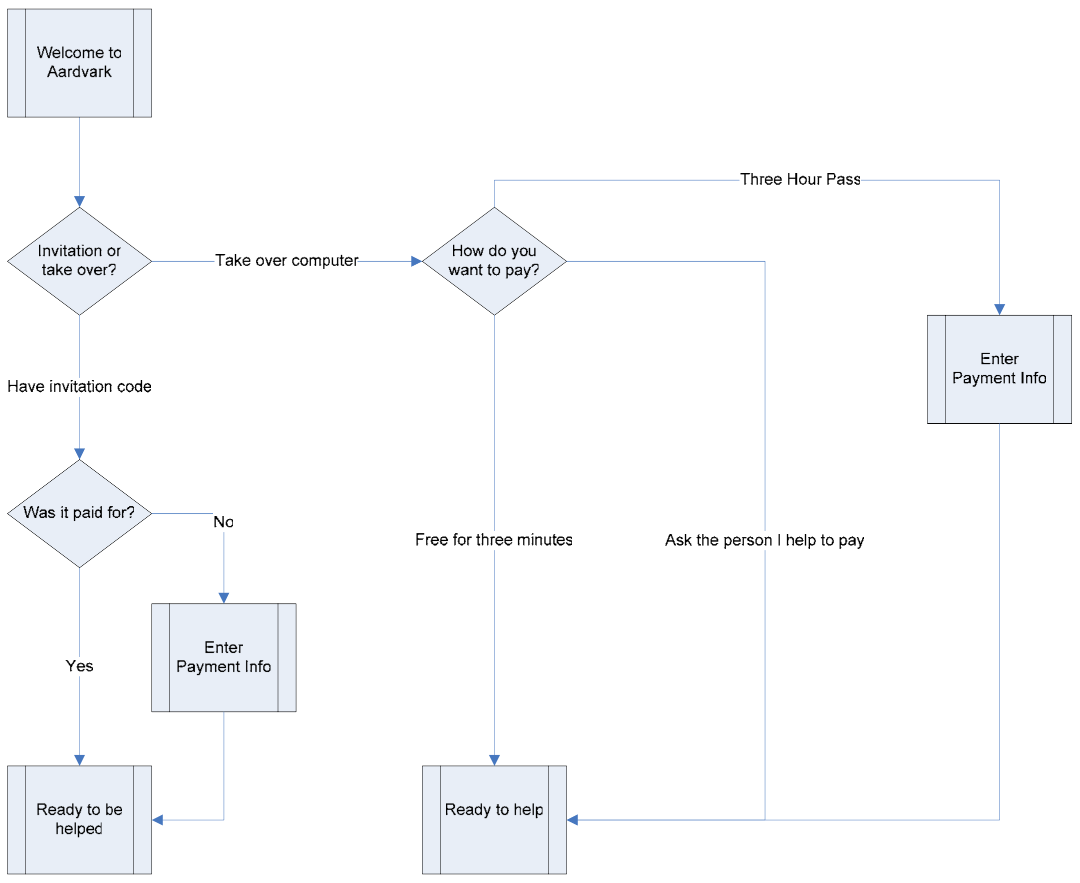
Implementation Note
The best way I can find to implement a self-deleting EXE is to extract a small deleter program to a temporary location using FILE_FLAG_DELETE_ON_CLOSE. This program waits for the current program to exit and then exits itself.

Future Features

- (hopefully v1) SSL encryption of all traffic, done entirely at the client to avoid massive CPU load on our servers.
- (v2) Reboot support, transparently. This works by having the victim client software detect a graceful Windows shutdown and put up a prompt on the screen asking if you want to reconnect on restart.
- (v2) Ability to transfer files between helper and victim, possibly with drag and drop
- (v2) Chat features, either voice or IM style
- (v2) Attempt to establish a direct connection first rather than going through the redirector if no firewall prevents it or if there is one unfirewalled direction between the helper and victim.
- (v2) Better firewall penetration by pushing VNC packets over HTTP requests, so even if they don't have access to port 80 or 443 but they do have an HTTP proxy to browse the web, we can blast through the firewall. (I don't know if this is necessary yet.)
- (v2) Record performance statistics and traceroutes, so in the future we can try to do some kind of performance optimization where we pick a reflector that will have the best performance for you. For example one day we might have servers in Australia and use them as the reflector when both the helper and victim are in Australia.
- (v2) Ship a shrinkwrapped, commercial reflector for tech support and large helpdesks.
- (v2) Improvements to the VNC protocol for performance and fidelity
- (v2) Digital signing of the downloaded EXEs to minimize scariness

Getting Started

Here's the general flow of the Aardvark website:



The Home Page

The branding of the home page is primarily Aardvark branding. Although there will be a small Fog Creek logo at the bottom, we really want people to think of Aardvark as it's own thing, not Yet Another Fog Creek product.

On the home page you see:

Welcome to Aardvark!

Aardvark allows people to help their friends, relatives, and customers fix their computer problems by temporarily controlling their computers via the Internet. It costs $7.95 for a day pass, or you can try it now, for free, for five minutes. Learn More!

To take over someone's computer:	To allow your computer to be taken over:
Your full name: [ ]	Invitation code: [ ]
[ ] I accept the Terms of Service	[ ] I accept the Terms of Service
[GO]	[GO]

Aardvark requires Microsoft Windows 95 or later.

Customer Service Brought to you by Fog Creek Software Privacy Policy and Terms of Service

The wording of this screen is likely to change. Helpers will enter their name and click "Take over someone's computer" while victims will enter an invitation code.

The full name is limited to 64 unicode characters.

We'll check the browser string to look for evidence of someone who is *not* on a Windows computer. It's not always possible to detect this. If we do find that they appear to not be on Windows, any attempt to go past this screen will display the following simple screen:

Aardvark requires Windows 95 or later on both computers. We were unable to verify that your computer is running Windows.

Yes, I am running Windows 95 or later.

No, I don't have Windows.

On the home page we'll set a test cookie just to make sure cookies are working properly. If the cookie doesn't arrive on the next page, you'll get this message:

Aardvark requires cookies to work correctly. They are essential so that we can get you reconnected if your connection is dropped.

Cookies appear to be disabled in your web browser. Please enable cookies and click here to continue.

The "terms of service" link pops up in a small external window.

If you fail to check the "terms of service" box, you are taken to a dedicated page which just asks, "Do you accept our terms of service?" with yes/no links. No takes you back to the home page.

The Learn More Page

The learn more page contains an overview and two separate FAQs, one for the helper and one for the victim. This page stresses simplicity, security, and ease of use. The FAQ may end up being a separate section, in which case it will be a link at the bottom of every page.

The Payment Page

The payment page looks like this:

How do you want to pay?

The Fog Creek Promise: If you're not satisfied, we don't want your money. You will not be charged until you are successfully connected.

Try Aardvark for free for 3 minutes

Ask the person I'm helping to pay

To purchase a 3 hour pass for $8:

[Paypal/Buy Now] ( ) MasterCard ( ) Visa ( ) American Express ( ) Discover

The first two options are not available in the case where the victim is being asked to pay because the helper already chose "ask the person I'm helping to pay."

- "Try Aardvark Free" takes you straight to the "Ready t0 Help" page.
- "Ask the person I'm helping to pay" takes you straight to the "Ready to Help" page, but issues an *unpaid invitation code*. When this unpaid invitation code is entered the victim is asked to pay.
- The Paypal link redirects you through Paypal for processing. This allows PayPal members to pay quickly without providing all their details. **Important Question – Can we authorize with PayPal and charge later, like credit cards? Otherwise we may have to refund to PayPal which would be a pain.**
- Clicking on one of the credit cards reveals a form with payment details:

How do you want to pay?

The Fog Creek Promise: If you're not satisfied, we don't want your money. Your card will not be charged until you are successfully connected.

Try Aardvark for free for 3 minutes

Ask the person I'm helping to pay

To purchase a 3 hour pass for $8:

[Paypal/Buy Now] ( ) MasterCard ( ) Visa (X) American Express ( ) Discover

Email Address: [ ]
We'll need this in case you get disconnected and need to reconnect, or if you need a refund for any reason.

Retype Email Address: [ ]

Card Number: [ ]

Expriation: (mm/yyyy) [ ]

Name, as it appears on card: [ ]

Billing address [ ]

Billing address (continued) [ ]

City [ ] State [ ] ZIP/Postal Code [ ]

Country [ ]

CVV: [ ]

[Click to pay with credit card]

This authorizes the credit card, but doesn't charge them. While the card is being authorized, they see this page:

> Your card is being authorized now (+ animation)

Implementation Note:

> This is implemented the same way as the shop. We simply make an entry in a database indicating that we need to charge the card. The HTML page saying "your card is authorizing now" is set to refresh every 5 seconds. In the meantime a service running on the SQL Server which does the actual authorization sees the new records in the database, authorizes the credit card, and writes the result back to the table. Eventually the HTML page is refreshed where it discovers that the card is accepted or declined, and it moves on to the appropriate page. The advantage of this method is that refreshing the browser is always safe and never double charges you.
>
> By the way, the invitation code is issued as an "unpaid invitation code" already on the "how do you want to pay," screen, where it's embedded as a hidden input field. This too is to prevent double-paying; if you submit, hit cancel real fast, edit something, and submit again, we can't double-charge you.

If the authorization fails, we take you back to the appropriate "how do you want to pay" screen, with a red box around the card number and an error message.

If it succeeds, we take you to the ready to help (or be helped) screen, as appropriate.

Ready to Help

The screen looks like this:

Ready to Help

Please tell the person whose computer you want to control to go to www.aardvark.com and enter invitation code **938 898 346 033**.

Then download and run this small program:

(icon)
Click to download.
(It takes about 30 seconds over a modem.)

The Aardvark Software is set up to connect directly to [counterparty]'s computer. When you are finished using it, it will delete itself from your computer.

Possible features of this screen:

- Sniff browser and give the user exactly correct instructions for downloading and running an EXE based on the type of browser they're connecting from
- The "30 seconds" figure needs to be calculated based on the actual experiences.
- Sniff browser to see if Java is going to work (possibly on an earlier screen) and use a Java client for the controller.

Ready to be Helped

Ready to be Helped

To allow your computer to be controlled by [counterparty's name], download and run this small program:

(icon)
Click to download.
(It takes about 30 seconds over a modem.)

The Aardvark Software is set up to connect directly to [counterparty]'s computer. When you are finished using it, it will delete itself from your computer.

Victim User Interface

The first thing the Victim sees when they run the EXE is a little window.

Let *helper name* control your computer?

[Yes] [No]

You will see everything they are doing, and you will be able to kick them off at any time.

Clicking No immediately exits the program and self-destructs.

Clicking Yes starts establishing the connection to the server. At that point the Victim only sees a very small tool window that says

```
Trying to connect.

[Exit]
```

Once the connection is established:

```
This computer is being
controlled remotely.

[Kick off helper]
```

The "Kick off" button goes to an "are you sure yes/no" message box just in case someone hits it by mistake. If you confirm that you want to click off the helper, the EXE exits and self-destructs.

The Victim User Interface does *not* use taskbar link like the listening VNC server, and it does not run as a server like traditional VNC.

Whenever the time remaining is less than 2 minutes, the UI changes:

```
This computer is being
controlled remotely.

[Kick off helper]

Time remaining: 1
minute 16 seconds
```

When the server is disconnected, the UI changes, depending on whether there is still time remaining or not. If no time is remaining:

```
Time up! This computer
is no longer being
controlled remotely.

[Exit]
```

If time is remaining, you go to the "reconnecting" message

```
The connection was lost.
Now trying to reconnect.

[Exit]
```

If we haven't been able to reconnect for 15 minutes:

```
The connection was lost.
Unable to reconnect.

[Try Again] [Exit]
```

Helper User Interface

The helper user interface is a modified version of the TightVNC viewer, with two major modifications. Dialogs will be removed, and the toolbar will be replaced.

All dialogs that allow the user to change settings will be eliminated, and the dialog that appears during the connection cycle will also be removed. The status area (described below) and hard-coded settings will take their place.

In place of the toolbar, a status bar shows text explaining what's happening. When the text changes, it flashes three times with a 1 second period so that you notice the change. Here are some messages you might see in the status bar:

> Trying to connect
> Controlling remote computer. Time remaining: 1 minute 18 seconds
> The connection was lost. Now trying to reconnect.
> The connection was lost. Unable to reconnect.
> Time up! You are no longer controlling the remote computer.

Invitation Codes

Design goals for invitation codes:

- easy to read over the phone
- easy for us to detect mistakes and typos when they are typed in
- do not reveal information about what other invitation codes might work

Here's the system. An invitation code is always twelve digits long and looks like this:

> 999 999 999 999

Spaces, dashes, and other punctuation are ignored on input, but we group the output into clumps of three to encourage people to read them to each other that way which should minimize mistakes due to faulty short term memory.

The invitation code will simply be 11 digits generated randomly using RNGCryptoServiceProvider. We try to insert this into a unique index column in the database; if that fails we try again with another random number up to 5 times; if that still fails we panic.

The twelfth digit is a checksum, calculated with mod 10 (LUHN), which is also stored in the database for convenience.

Reconnecting

If a user pays for a pass but then manages to close their web browser, or otherwise get disconnected, they won't want to pay again.

When they come back to www.aardvark.com, we check their cookie, a unique ID that consists of an invitation code plus a code saying whether they are the helper or victim.

If a helper comes back while there is still time remaining on the invitation code, they will see a slightly modified home page with a prominent "resume" option and "someone" changed to "someone else"

> Welcome to Aardvark!
>
> **Did you get disconnected? Click here to resume helping the last person you were helping.**
>
> Aardvark allows people to help their friends, relatives, and customers fix their computer problems by temporarily controlling their computers via the Internet. It costs $8 for a three hour pass, or you can try it for free for three minutes.
>
> Do you have an invitation code? Enter it here: [ ] [Go!]
>
> Or, enter your full name to take over someone else's computer: [ ] [Go!]
>
> Important! Aardvark requires Microsoft Windows 95 or later.

And of course that just takes you back to the helper download page.

If a victim comes back while there is still time remaining on the invitation code, they still see the home page, but the invitation code is already filled out. Thus clicking on the **Go** link takes them right to the download page and they're all set. They also see the message:

> Welcome to Aardvark!
>
> **Did you get disconnected? Click here to let John Smith control your computer again.**
>
> etc…

These "Did you Get Disconnected" messages actually look like post-it notes and they obscure the text below them, so nobody will miss them. They have close buttons, handled with JavaScript.

How Sessions Are Timed

The server starts the countdown clock from the time that the first connection is successfully established.

If the user deliberately disconnects, the clock keeps running.

If the connection is severed for some reason, the clock stops until the connection is resumed.

We could simplify this dramatically by using longer passes (8 hours or 1 day passes) so that people are less likely to fret about the occasional outage. That way we don't need to ever stop the clock.

Data Collection

(low pri) The reflector will keep track of how many bytes it reflected for each invitation in a 64 bit int. We will use this to keep track of our bandwidth costs.

Logging

The reflector and web site will log all significant events in a table, indexed by invitation code and with an accurate time stamp. This way we can quickly see the history of any session for diagnostic purposes. Things which will be logged include:

- New invitiation code created
- Helper elects 3 minutes free
- Helper elects to pay, with payment info, including only the last 4 digits of the credit card (or the last 5 digits if AMEX)
- Helper asks victim to pay
- Helper elects to pay with PayPal
- PayPal approves transaction
- Credit card co has authorized transaction
- Helper got to download page and started downloading
- Victim got to download page and started downloading
- Helper successfully connected to reflector
- Victim successfully connected to reflector
- Card was finally charged
- All steps in the negotiation protocol
- Helper or Victim connection lost
- Helper or Victim forced off because counterparty lost connection

Conflicts and Communications Problems

To insure a seamless experience this section will list possible things that can go wrong and explain how we recover from them gracefully.

Problem	**How it will be handled**
Software can't connect to the reflector because of a DNS problem.	The helper/victim software knows the absolute IP address of the reflector it's supposed to use and never uses DNS.
Software can't connect because they no longer have Internet connectivity.	Display message, keep retrying for 15 minutes. After 15 minutes give up. **(v2)** Start a thread that tries to download a simple test page from our web site using IE. If that succeeds but our connection fails, assume that we're being blocked by firewall and display a message to that effect.
Connection drops (one client)	Reflector forcefully drops the connection to the *other* client just to make this the same as the next case:
Connection drops (both clients)	Client displays a message saying "Connection lost, trying to reconnect" and forces its window onto the top of the screen (yeah people love that) Start again with the protocol from the beginning. Keep retrying for 15 minutes. After 15 minutes give up.
One of the clients loses their EXE, but they are still on the Aardvark web page	They redownload it and all is well
Victim loses their EXE, and has closed their web browser.	Victim goes back to Aardvark.com and sees their invitation code already typed in. If that doesn't work because cookies are disabled or getting deleted too often, he asks the helper for the invitation code again.
Helper loses their EXE and has closed the web browser.	Goes back to Aardvark.com and sees the special link to resume the previous session on which time remains.
Helper loses their EXE, has closed the web browser, and deleted their cookies.	They'll have to contact us for a refund.
One of the clients connects to the reflector, then *another* client of the same type connects to the reflector before any session is established.	First client is disconnected with a GOAWAY ANOTHER message.
While a session is in progress, another helper (victim) comes along.	Second client is disconnected with a GOAWAY INPROGRESS message.

A session is hung or lost somehow. It's not working, but both the helper and victim are somehow talking to the reflector. Additional clients try to connect but they keep getting GOAWAYINPROGRESS's	I'm not sure how this could happen, but I suspect it's possible. They will have to relaunch, pay again, and get a refund.
Helper and victim happen to be connected to different servers by the load balancer	When the first party connects, the IP address of the machine they connected to is stored in the database. When the second party connects, if they have connected to a different server, they are given a BOUNCE message containing the server where the first party is waiting for them.

Security

We will modify VNC to use SSL instead of plain sockets for communication. The SSL session will be end-to-end, in other words, it will be between the two clients, without the reflector being involved at all. In fact the reflector just passes bytes back and forth which happen to be encrypted. This design is intended to maximize security and to let the CPU-intensive key generation happen on the clients, not on our servers.

That means that we can't *really* snoop on the VNC traffic to decide when to charge the credit card. We'll have to look more closely at the protocol to decide if there is some point at which we can be fairly sure that the connection is going both ways…maybe a certain number of bytes going each way.

The Aardvark website is an SSL website from top to bottom. It may, some day, need an SSL accelerator.