Poseidon House
Castle Park
Cambridge CB3 0RD
United Kingdom

TELEPHONE: **Cambridge (01223) 515010**
INTERNATIONAL: **+44 1223 515010**
FAX: **+44 1223 359779**
E-MAIL: **apm@ansa.co.uk**

ANSA Phase III

The WWW Project: ISF Briefing to IBM 4/9/95

Mark Madsen

Abstract

This is a briefing presentation given to visitors from IBM Hursley on the occasion of their visit to the Information Services Framework Group at APM in Cambridge on Monday the 4th of September 1995.

The slides describe the ISF Group's project to re-engineer the WWW infrastructure to incorporate support for distributed object technologies. The commercial and technical motivation for undertaking such work is oulined, followed by a summary of the results from first two phases of the project.

APM.1561.00.01 **Draft** 1st September 1995

Briefing Note

Distribution:
Supersedes:
Superseded by:

The WWW Project:

Re-Engineering the Web with Objects

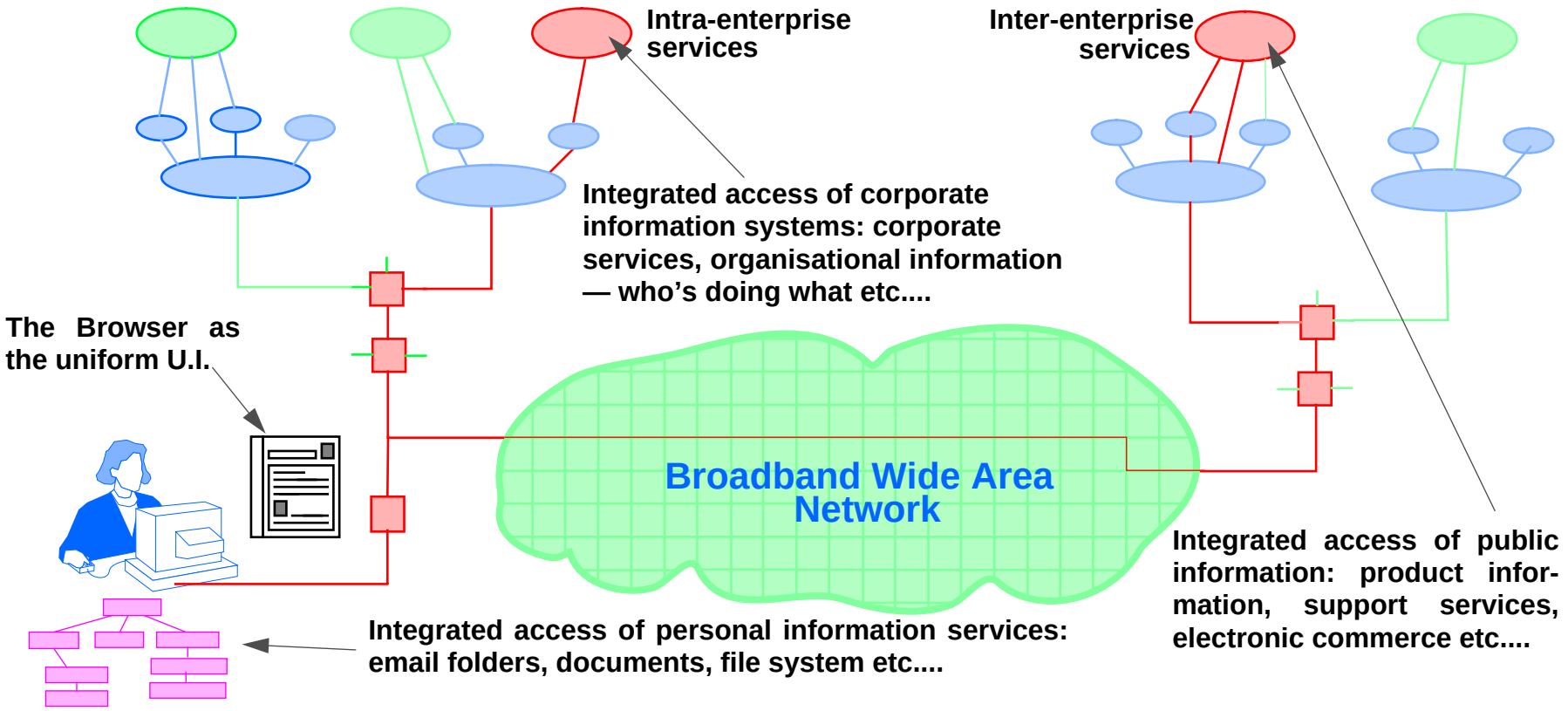
Information Services Framework Group
ANSA Project Phase III

Presented by Mark Madsen

Introduction

- How WWW is creating a “Uniform” information space
- What technology is needed?
- How distributed objects can help
  - What we have done
  - What we are doing

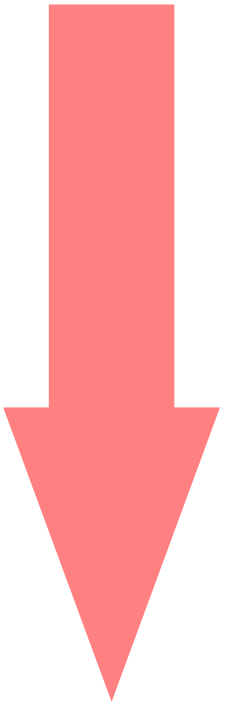
Creating a uniform information space



Technology requirements for the uniform information space

Well understood

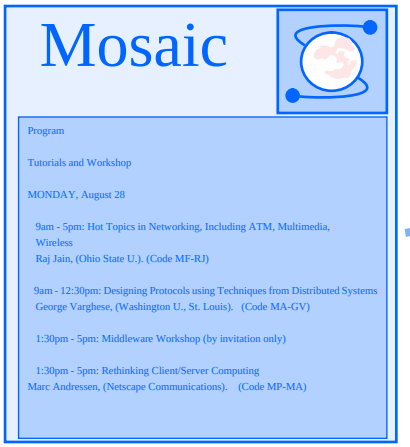
- Presentation (= Browsers + HTML, postscript etc.)
- Creation (= Authoring tools)
- Efficient protocols
- Extensible front ends (making new functionality available)
- Extensible back ends (integrating new & existing systems)
- Navigation tools (finding the information)
- Administration tools (managing the services)



Poorly understood

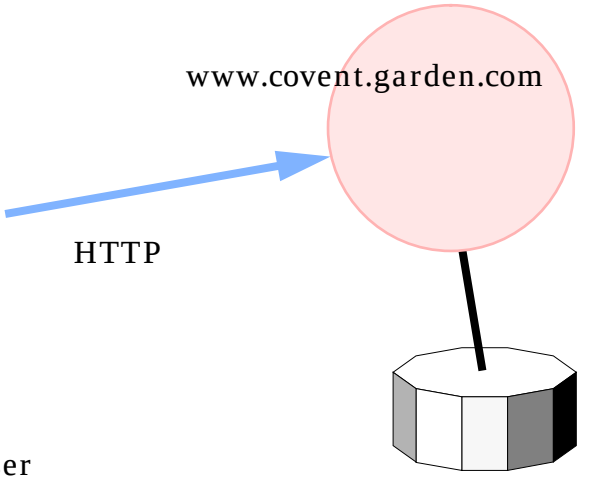
A “commercial” application

- **Scenario: booking theatre tickets for multiple events.**
- **Requirements**
  - **Browsing program information**
  - **Choice of seats & dates**
  - **On-line selection of preferences**
  - **Confirmation of booking by server**
  - **Ability to change booking**
  - **Single payment on confirmation of booking by client**



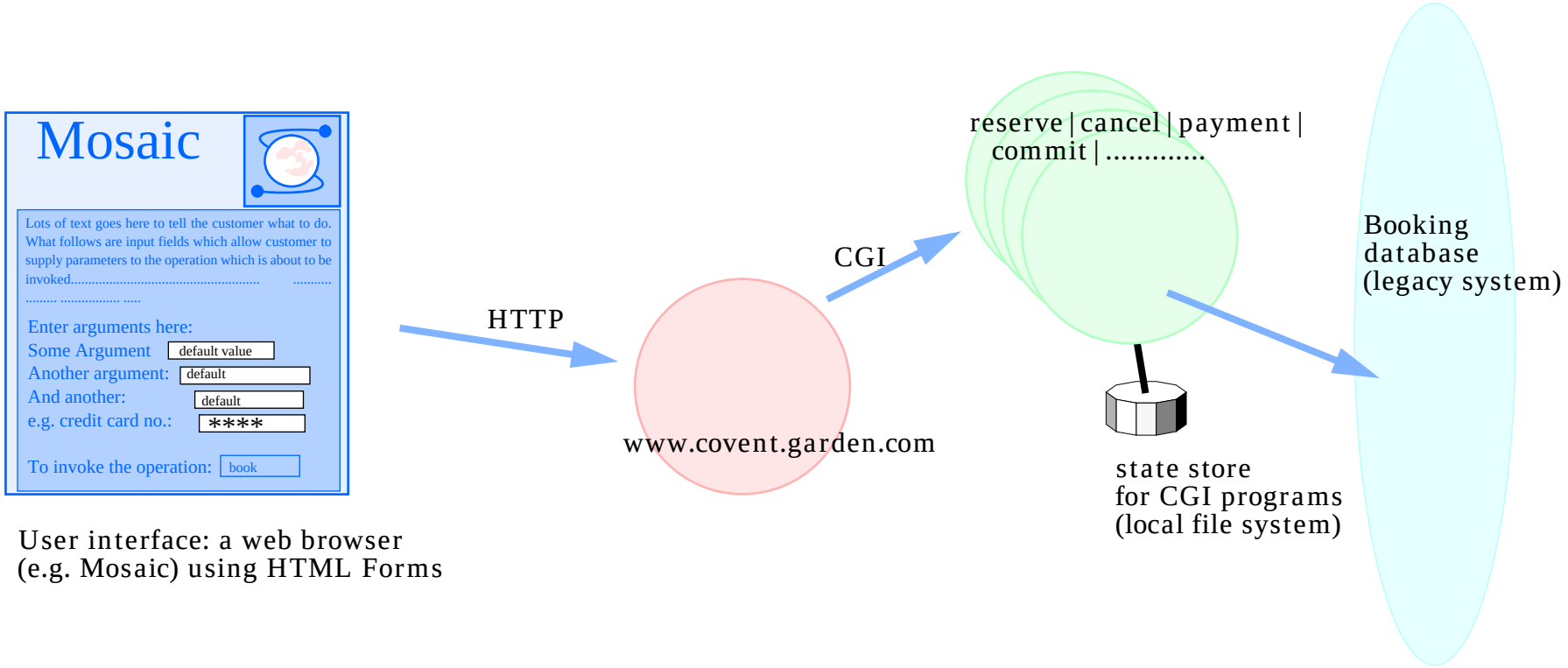
User interface: a web browser (e.g. Mosaic)

HTTP



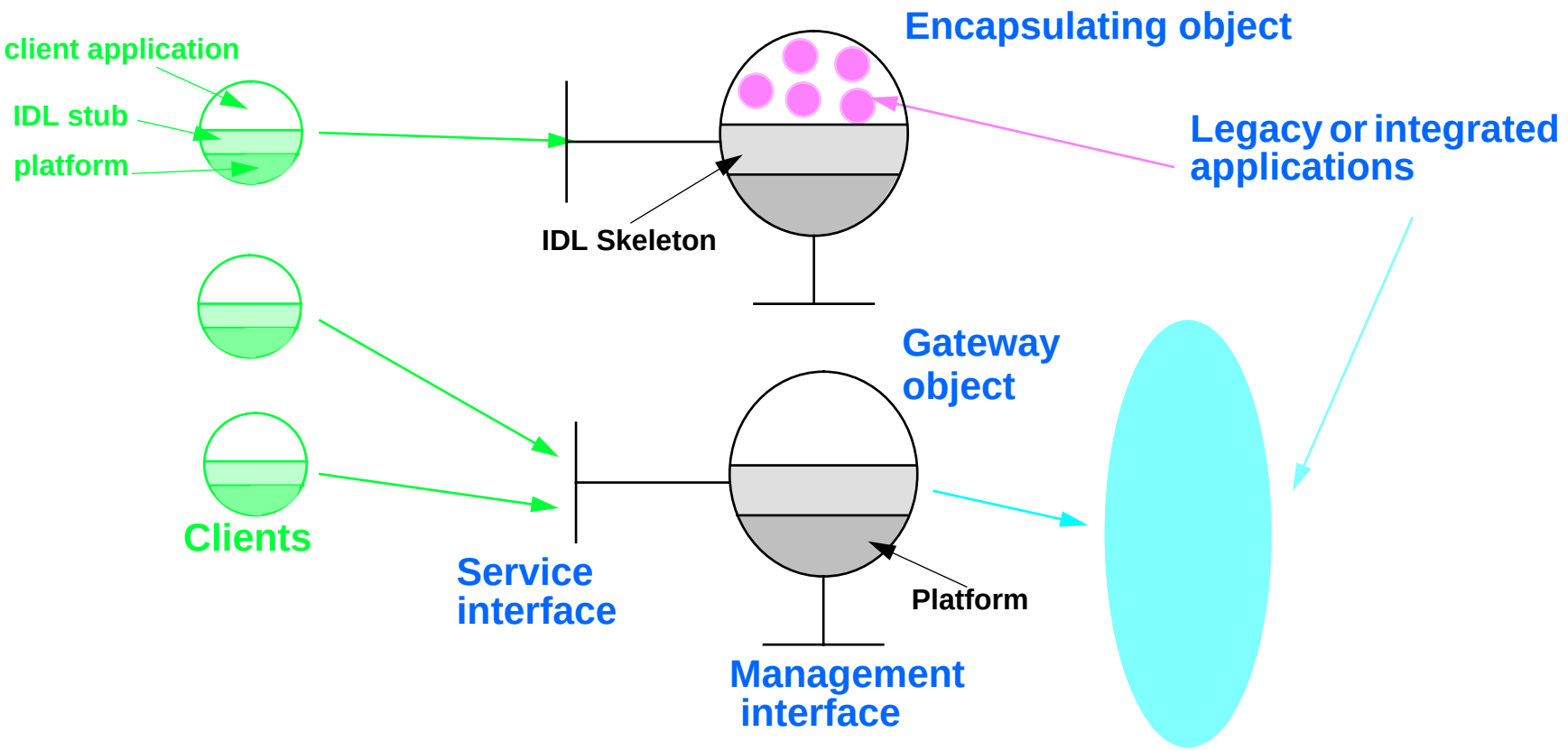
Document store containing program information

Extending the functionality of WWW using CGI Programs



User interface: a web browser
(e.g. Mosaic) using HTML Forms

The Solution: Distributed Object Technology



ANSAweb Phase 1: A Stub Compiler for the web

Browser

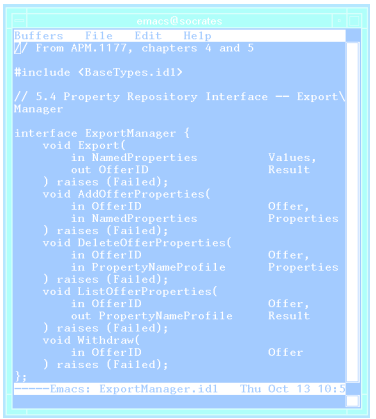
HTTP Server

Application

IDL Skeleton

CGI program

CORBA IDL for Service



Generate
Form (on the fly)

Generate
IDL stubs for clients

Generate IDL
skeletons for servers

IDL stub
libwww

Bespoke client

Example: HTML Form Generation

```
interface Echo{
   string Echo(in string Src);
   void Sink(in string Src);
   string Source(in long Length);
   string Reverse(in string Src);
};
```

Stub Compiler generates this HTML form

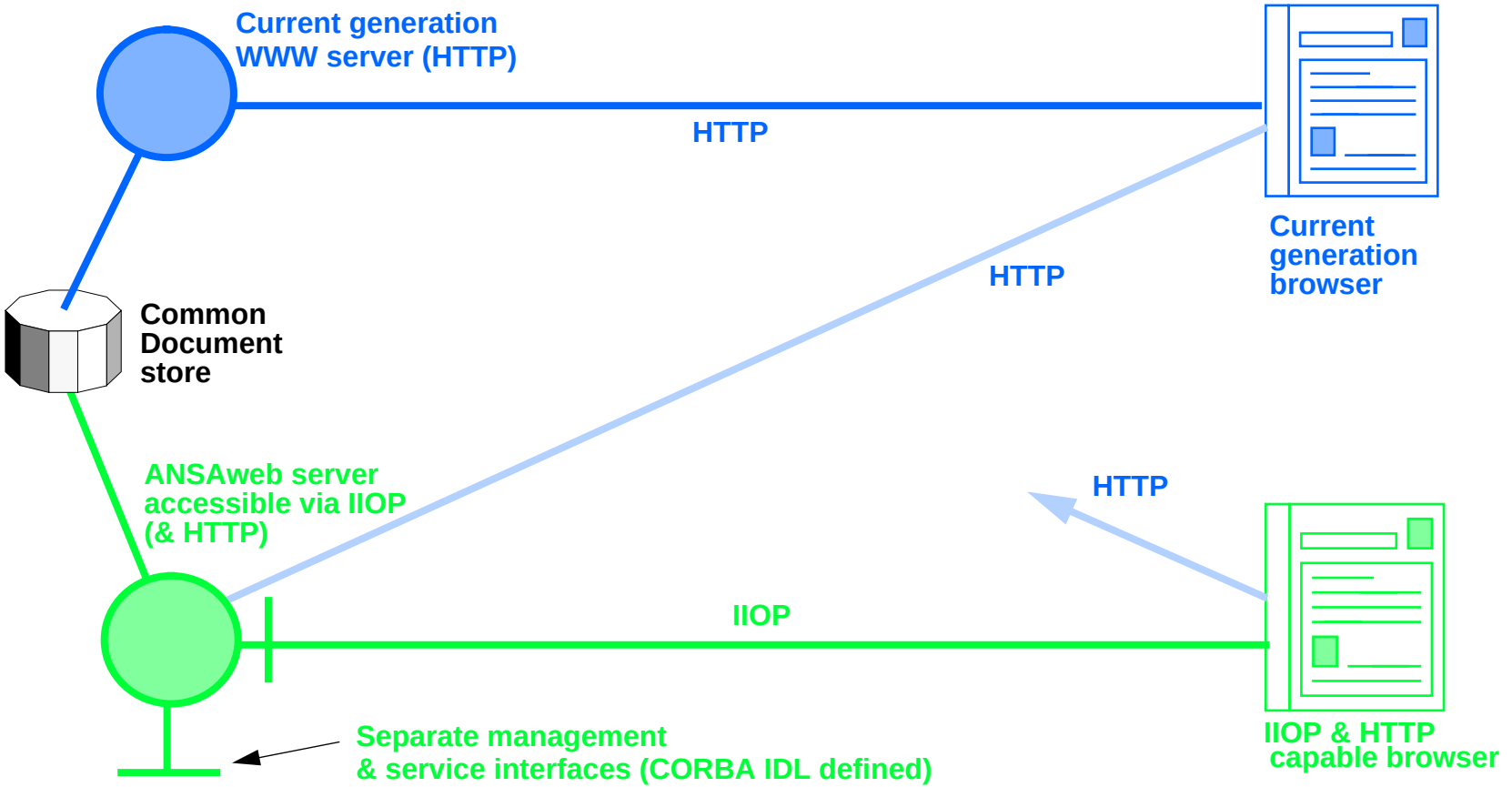
```
<head>
<TITLE>Input for Echo</TITLE>
</head>
<BODY><H1>Input for Echo</H1>
<HR>

<H2> Operation Echo</H2>
<FORM METHOD=”POST” ACTION=”http://socrates.ansa.co.uk:8080/cgi-bin/Echo”>
<P>Enter arguments here:<P>
<INPUT NAME=”Operation” TYPE=hidden VALUE=”Echo_Echo”>
CORBA_string Src: <INPUT SIZE=10 NAME=”Src”> <P>
To invoke Echo_Echo: <INPUT TYPE=”submit” VALUE=”Echo_Echo”><P>
</FORM>
<HR>

<H2> Operation Sink</H2>
<FORM METHOD=”POST” ACTION=”http://socrates.ansa.co.uk:8080/cgi-bin/Echo”>
<P>Enter arguments here:<P>
<INPUT NAME=”Operation” TYPE=hidden VALUE=”Echo_Sink”>
CORBA_string Src: <INPUT SIZE=10 NAME=”Src”> <P>
To invoke Echo_Sink: <INPUT TYPE=”submit” VALUE=”Echo_Sink”><P>
</FORM>
<HR>

<H2> Operation Source</H2>
<FORM METHOD=”POST” ACTION=”http://socrates.ansa.co.uk:8080/cgi-bin/Echo”>
<P>Enter arguments here:<P>
<INPUT NAME=”Operation” TYPE=hidden VALUE=”Echo_Source”>
CORBA_long Length: <INPUT SIZE=10 NAME=”Length”> <P>
To invoke Echo_Source: <INPUT TYPE=”submit” VALUE=”Echo_Source”><P>
</FORM>
<HR>

<H2> Operation Reverse</H2>
<FORM METHOD=”POST” ACTION=”http://socrates.ansa.co.uk:8080/cgi-bin/Echo”>
<P>Enter arguments here:<P>
<INPUT NAME=”Operation” TYPE=hidden VALUE=”Echo_Reverse”>
CORBA_string Src: <INPUT SIZE=10 NAME=”Src”> <P>
To invoke Echo_Reverse: <INPUT TYPE=”submit” VALUE=”Echo_Reverse”><P>
</FORM>
</BODY>
```

ANSAweb Phase 1 — Benefits

The benefits arise from having to write less code

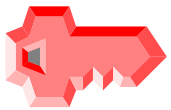
- **Productivity**
  - It is easier to interface existing systems into WWW
  - Do not require a deep understanding of platforms and protocols
  - Remote invocations look like local invocations
- **Less errors**
  - Template forms are correct HTML
  - Template forms, stubs and skeletons are consistent
- **Protection against changes**
  - Skeletons and stubs abstract the programmer from underlying platform — if the platform or protocol changes change the stub compiler and regenerate the stubs, do not have to rewrite the application.

ANSAweb Phase 2: Migrating the Web to Distributed Objects



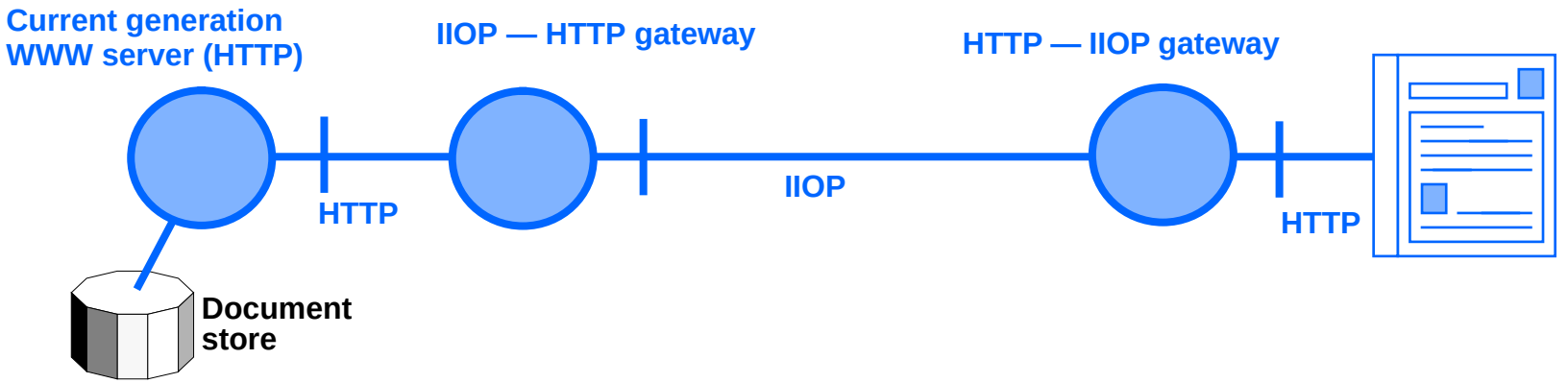
The benefits

- Performance and capabilities of IIOP
- Extensibility
  - The ANSAweb server is essentially a CORBA object
  - The ANSAweb server gateway to the CORBA world
  - Interfaces defined and extensible in IDL
  - Integrating third party systems is a strength of CORBA
- Clients (browsers) migrate to object technology
  - Enhancements to WWW clients become simpler
- Manageability



Backwards compatibility — all existing WWW functionality and resource investment is preserved

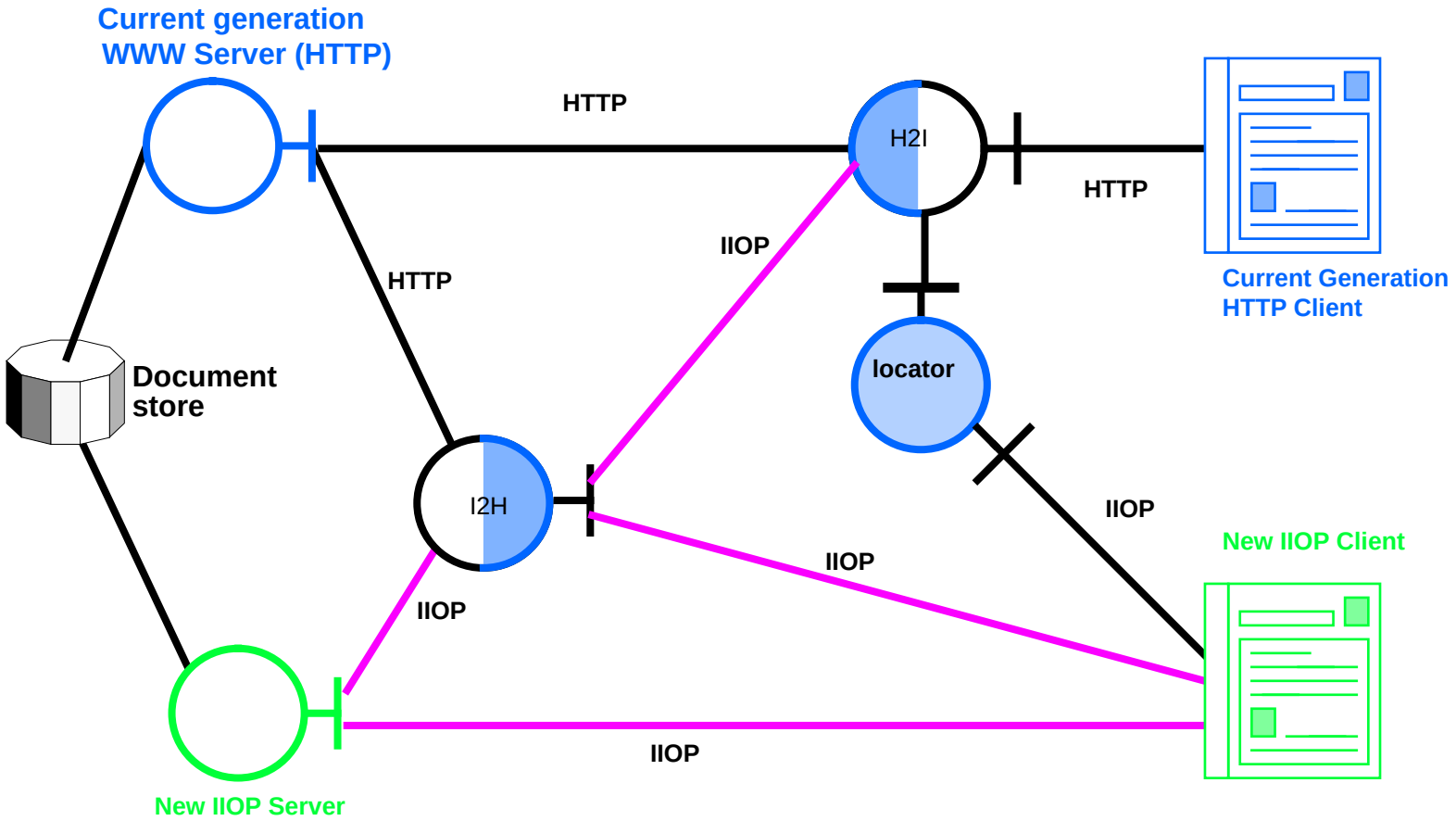
Getting to the promised land — the intermediate step



Status

- Initial versions of the intermediate components working
- IIOP implementation is the Sun public domain code
- An IIOP stub compiler has been built using the Sun public domain front end

Native IIOP and Locator Technology



Status of Native IIOP Componentry

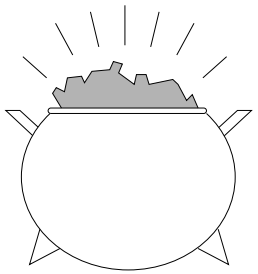
- Server functionality has been built using different technologies
  - Sun's Inter-Orb Engine
  - Iona's Orbix
- Initial versions of the native IIOP clients are working:
  - Line Mode Browser (Common Code Library test client)
  - Arena (W3C's HTML-3 testbed graphical client)
- Other clients can easily be modified to use ANSAweb components
  - Main problem is profusion of WWW Library versions
- The basic Locator is in place
  - This will ultimately be complemented by a fully capable Trader

Conclusions

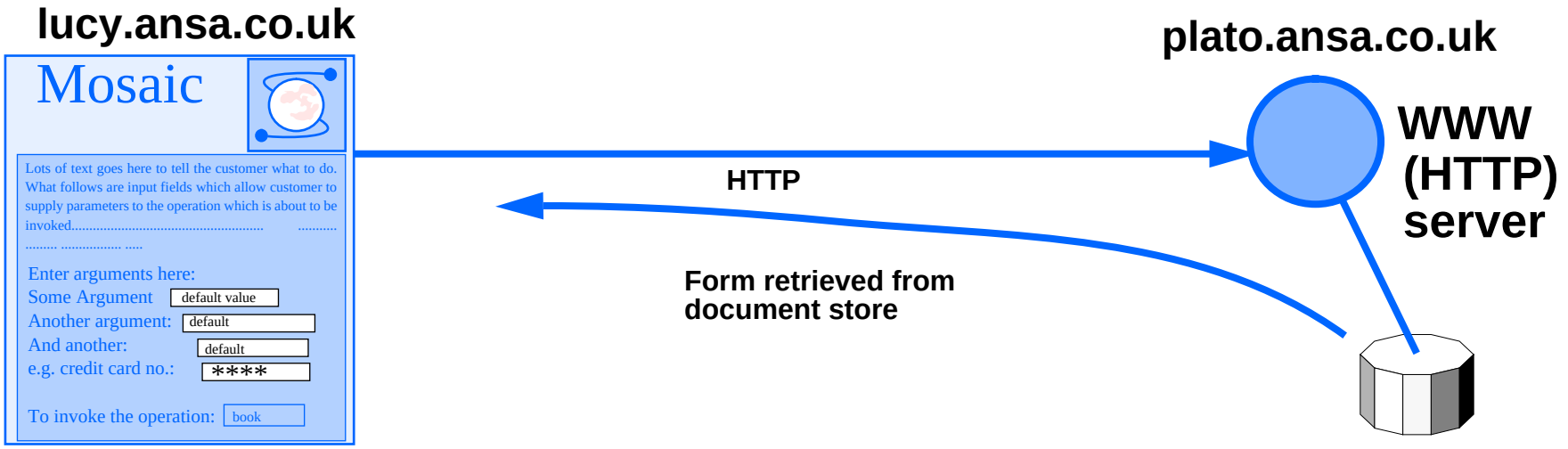
- The web is creating a uniform information space
- Some technology still required: efficient protocols, extensible front and backends, navigation tools, administration tools
- The distributed objects apprach offers a solution
- The benefits include: programmer productivity; fewer programmer errors; hiding of changes in the underlying platforms and protocols; extensibility (ease of integrating new and existing systems); performance of the underlying protocols
- ANSAweb Phase 1 released June 9 1995 to establish a presence
- ANSAweb Phase 2 to be announced in December 1995 at the WWW conference

Introduction to the Simple Bank

- The Simple Bank is a generic ANSA demonstration, because it incorporates many aspects of distributed object technology in a simple, real-world-connected example:
  - dynamic interface creation
  - interface reference passing
- The Simple Bank offers a single service with multiple interfaces
- Login corresponds to creating an interface
- Management and customer interfaces each maintain state at each end of the interface

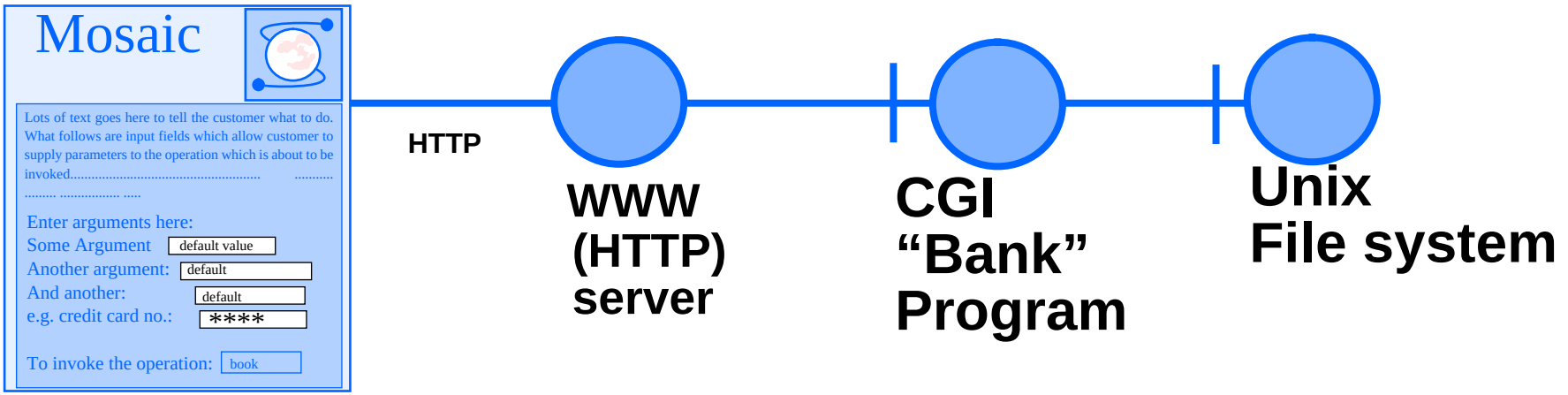


The simple bank demonstration



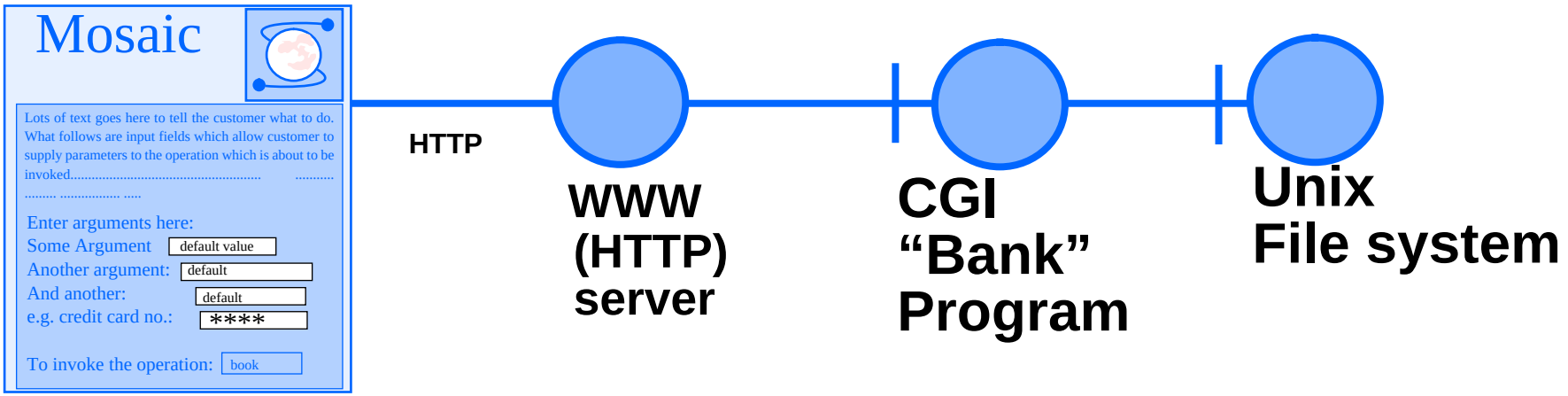
- The template form was generated by the stub compiler
- This was edited and loaded into the document store
- Comparison of template and edited form is instructive

The Simple Bank Manager



- The manager fills out the form interface to manage the bank
- HTTP server launches the bank program
- State is stored on the file system
- Result returned as HTML
- CORBA Exception Handling increases robustness

The customer



- The customer fills out a form supplying PIN and Account number
- HTTP server launches the bank program
- Bank program checks the PIN and Account number are valid and generates a UI for the customer to access their account ONLY
- UI is a modified template form generated by stub compiler
- It is generated by the bank program on-the-fly, rather than being stored in the file system

Lessons

- The application was developed in about 10 working hours
- 500 lines of C (includes an edited form generated by the stub compiler)
- Most of the complexity was in manipulating the unix file system to create a persistent state store and also setting file system locks to deal with concurrent access (most of the code was needed for this)
- Editing of forms takes only a few minutes
  - It is worth while running them through an HTML checker to make sure they are still correct after editing
- Experience suggests this is a useful tool for interfacing legacy systems into the web.