JSPServlet MAPPER

A PageBox implementation

1 Definitions

1.1 Presentation

We define a presentation here as a set of components able to generate and format what is displayed on a user equipment.
A presentation calls content producers to get the content that it processes.

A presentation as described in this document handles only HTTP/HTTPS though an intermediate gateway can convert from HTTP to another protocol such as WAP. A presentation can generate a flow in miscellaneous formats such as XML, HTML, XHTML, WML, PDF, SVG and SWF.

A presentation calls a content producer using a network protocol such as XML over HTTP, SOAP or RMI. It can have a function of content adaptation.

A presentation can be implemented as a J2EE Web Archive.

1.2 PageBox

PageBox is a mean enabling the hot deployment and update of presentations in Application Servers.
A PageBoxed Application Server (PAS) behaves as a browser, it downloads a presentation just like a browser downloads an applet and like a browser a PAS runs the presentation in a sandbox with rights based on the presentation signature.

Unlike a browser, a PAS downloads or updates a presentation only when commanded through an HTTP request.

PageBox has been designed for Internet deployment and for Application Service Providers (ASPs). It uses only HTTP/HTTPS protocol and allows a simpler and safer deployment than FTP.

1.3 Publish and Subscribe

A PageBox can subscribe to one or many repositories.

When a new archive is published on a repository, it is automatically deployed on all PageBoxes that have subscribed to the repository.

1.4 Constellation

A constellation is a set of PageBoxes and repositories with a trust relationship. Whereas PageBoxes and repositories have a physical existence (they are hosted somewhere and run PageBox code) a constellation is an organizational entity. A constellation can be characterized by:
?? A list of repositories
?? A common security mechanism allowing subscription and publication checking

1.5 Mapper

The mapper is a mechanism embedded in PageBox that modifies links in specified static pages (HTML, XHTML), named routing pages.
A user enters a presentation when it specifies a well-known URL, handled by one PageBox or a small set of clusterized PageBoxes.

When the user clicks on a routing page, links can be modified to:
?? Select a PageBox closer to the user
?? Load-balance requests between PageBoxes

2 Executive summary

PageBox infrastructure is an example of distributed infrastructure without central point of administration.

The infrastructure includes:
?? PageBoxed Application Servers (PAS) that run presentations
?? Presentation repositories managed with publish and subscribe

An entity – for instance an ASP – can install a set of PageBoxes and subscribe them to a set of repositories. Another entity – for instance a software company that writes presentations – can create a repository.

Thank to Mapper users don't need to know which PageBox they should use – presumably the closest -. They always enter the same URL and Mapper routes them automatically to the best Presentation site.

The infrastructure is fault-tolerant:
?? If a PAS crashes, requests are balanced between remaining PAS
?? If a repository is unavailable, it simply means publication and subscription won't be possible up to the time the problem is fixed
?? If a PAS lose its data, it simply need to subscribe again

3 Foreword

The Mapper relies on other infrastructure parts:
?? PageBox subscription
?? Archive publication
?? Resource handling in PageBox

Therefore it requires no customisation of PageBoxes and repositories.
The Presentation provider (aka the publisher) has only to understand how to specify a routing page.

3.1 Parameterisation

A routing page is a HTML or XHTML static page with <A HREF=…> included in the Presentation archive and listed in a RoutingPages.properties.

The RoutingPages.properties file must stored at the root of the Presentation archive and contain lines in format Routing_page_path = comment where Routing_page_path is the full path of the routing page in the archive. For instance:

MyDirectory/MyPage.html = MyComment

3.2 Use

An archive designed for routing should include only relative URLs <A HREF="xxx" …> but not <A HREF="/xxx" …> nor <A HREF="http://xxx" …>.
PageBox changes only <A HREF="xxx"…> links in routing pages and only when the user is not in session.

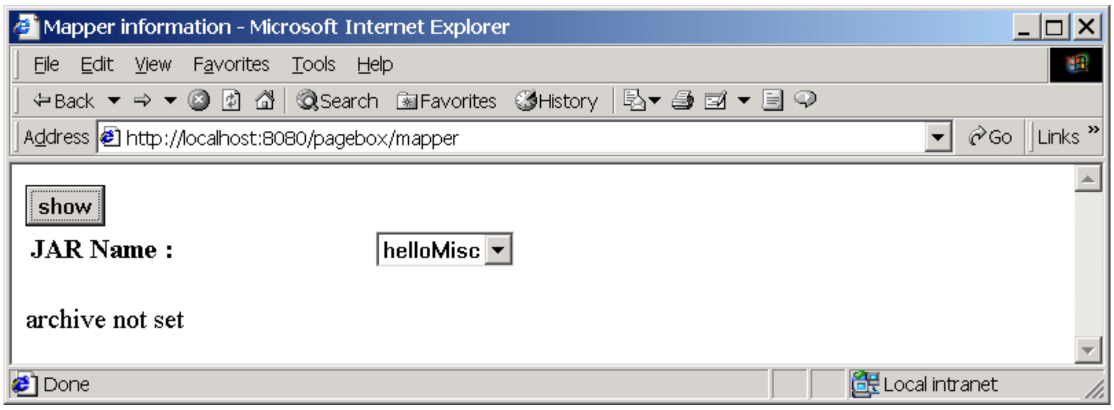
We recommend declaring in RoutingPages.properties only your site entry pages (the pages you advertise and that are referenced by search engines).

3.3 Content

Therefore this document focuses on the implementation in order to simplify troubleshooting and to explain how to write a mapper. It is divided in four sections:
?? The monitoring URL
?? Publish and subscribe
?? Mapper
?? Routing page handling

4 Monitoring URL

The monitory URL is embedded in JSPservlet, so it is not a servlet.
You invoke it with PageBoxURL/mapper:

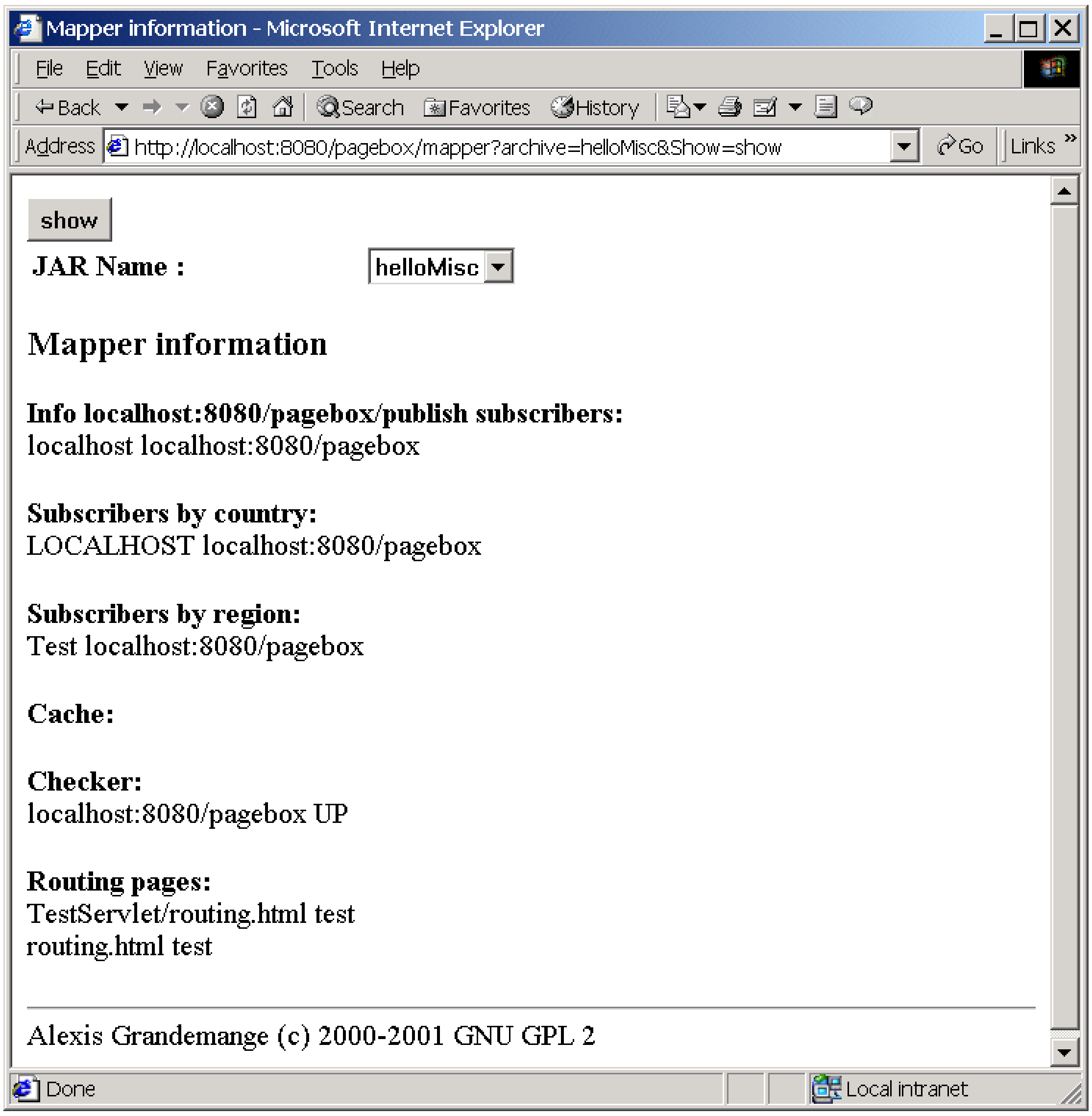

Mapper information - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://localhost:8080/pagebox/mapper
Go Links
show
Done
Local intranet

JAR Name : helloMisc

archive not set

Figure 1: mapper first screen

The drop-down list contains all published archives.
Selects the archive you want to monitor in the list and click show. You are displayed the report below:


Mapper information - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://localhost:8080/pagebox/mapper?archive=helloMisc&Show=show
Go Links
show
Done
Local intranet

JAR Name : helloMisc

Mapper information

Info localhost:8080/pagebox/publish subscribers:
localhost localhost:8080/pagebox

Subscribers by country:
LOCALHOST localhost:8080/pagebox

Subscribers by region:
Test localhost:8080/pagebox

Cache:

Checker:
localhost:8080/pagebox UP

Routing pages:
TestServlet/routing.html test
routing.html test

Alexis Grandemange (c) 2000-2001 GNU GPL 2

Figure 2: mapper report screen

It displays:

- Subscribing PageBoxes
- Subscribing PageBoxes per country and per region
- Cache entries. None on the screen shot. In format first IP address – last IP address PageBox list
- State of PageBoxes UP or DOWN
- Archive routing pages

5 Publish-subscribe

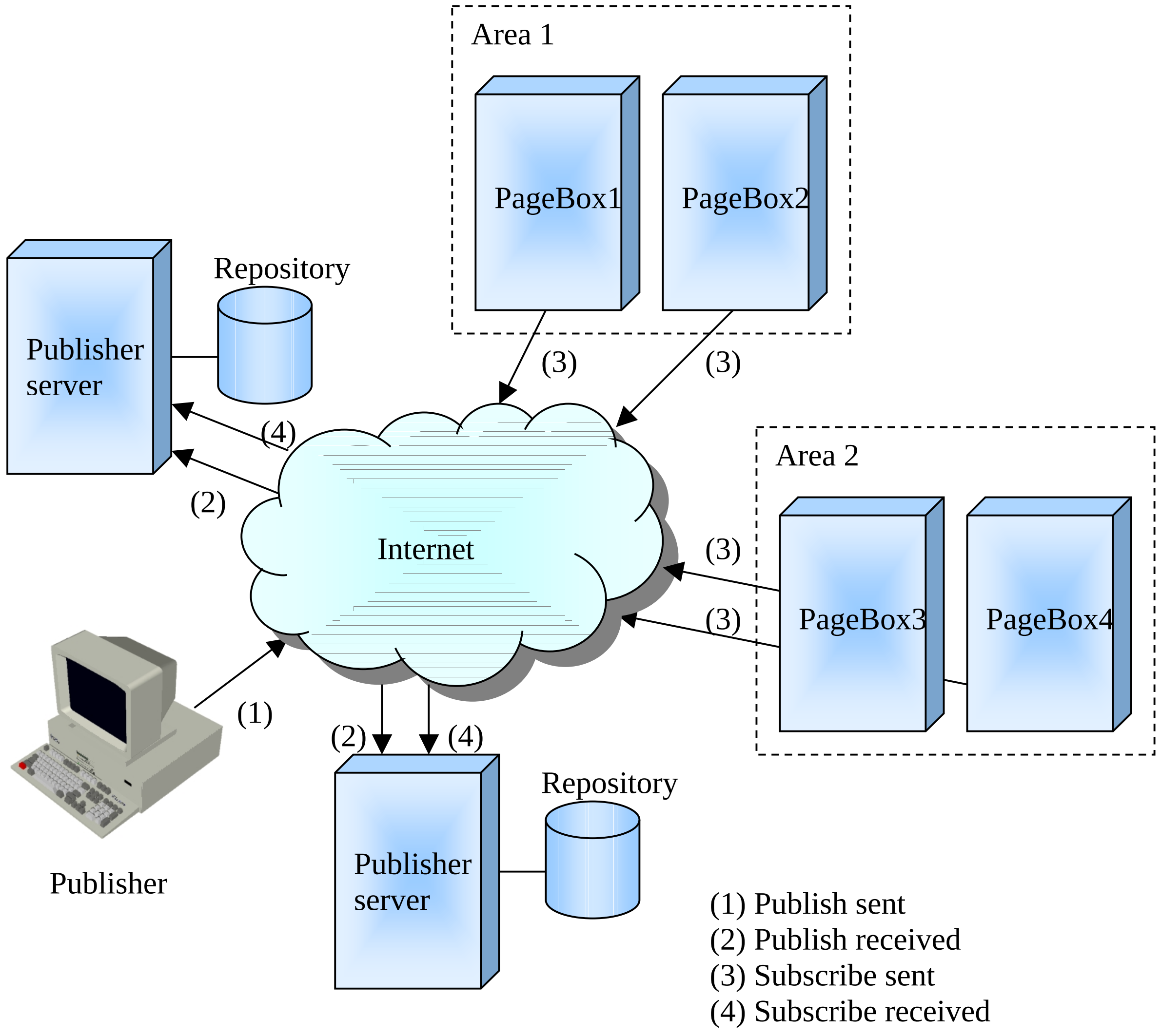


Figure 3: Publish-subscribe

Entities that want to deploy a presentation:

1. Package the presentation in an archive
2. Publish the archive in a repository using the PublisherClient tool

The PublishServer that manages the repository deploys the archive on subscribed PageBoxes. Note that PageBoxes are not necessarily aware they belong to the same area: the Mapper establishes this classification. Note also that a PageBox can subscribe to many repositories. In that case the PageBox hosts archives from different sources.

6 Mapper

6.1 Protocol

The infrastructure involves 3 entities:

?? PublisherServer that manages the repository and receives publication and subscription requests

?? PageBox that hosts the presentation and the mapper

?? The requestor

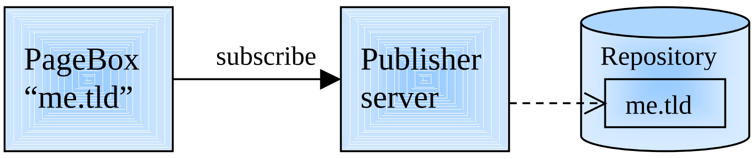


Figure 4: step 1 – subscription

The PageBox owner subscribes on PublisherServer. The PublisherServer registers the subscription in the repository.

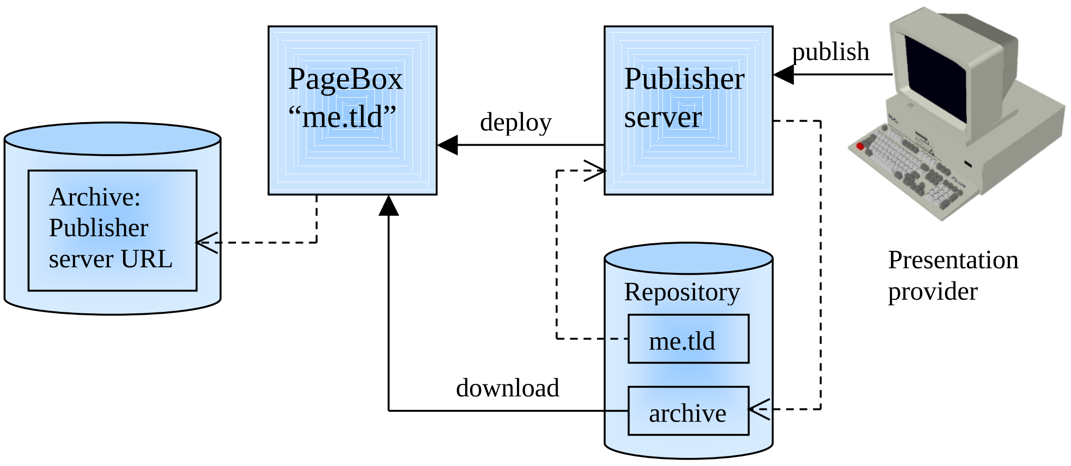


Figure 5: step 2 - publication and deployment

When an entity publishes an archive, it uploads the archive and the PublisherServer stores it on the repository. Next it sends a deploy command with its own URL to its subscribers.

When a PageBox receives a deploy request, it:

1. Records the archive was deployed by this PublisherServer
2. Downloads the archive from the repository

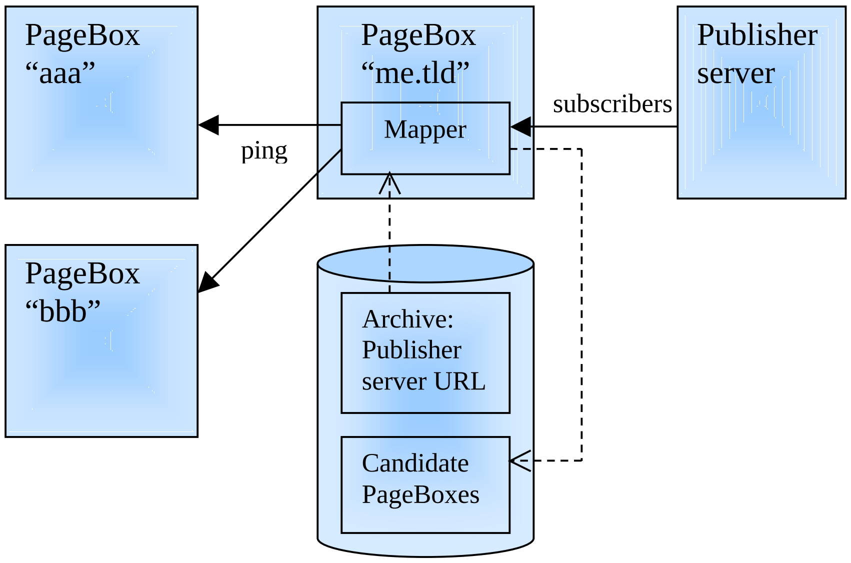


Figure 6: Mapper initialization and ping

When Mapper initialises, it downloads the subscriber list from the PublisherServer URL. Then it "pings" the other subscribing PageBoxes to check if they run. Note that it uses an HTTP ping handled by PageBox code.

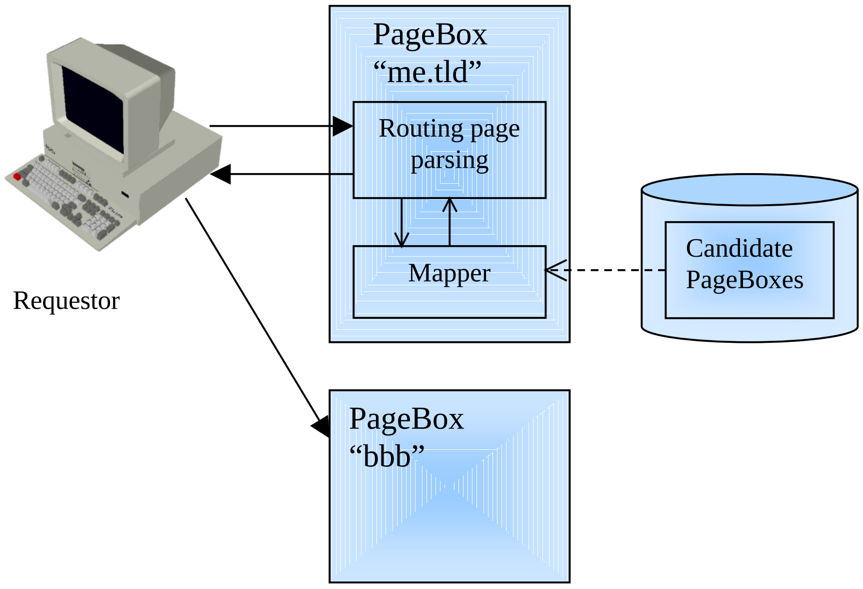


Figure 7: step 3 - routing

When a user requests a routing page:
1. PageBox calls Mapper with the requestor host name
2. Mapper analyzes the host name and find the closest PageBoxes
3. It selects one of them using a round-robbin algorithm
4. PageBox updates the links and returns the page

Then when the user clicks on a link, it is routed to a closer PageBox.

6.2 PageBox selection

To illustrate this section we show two worldwide constellations.

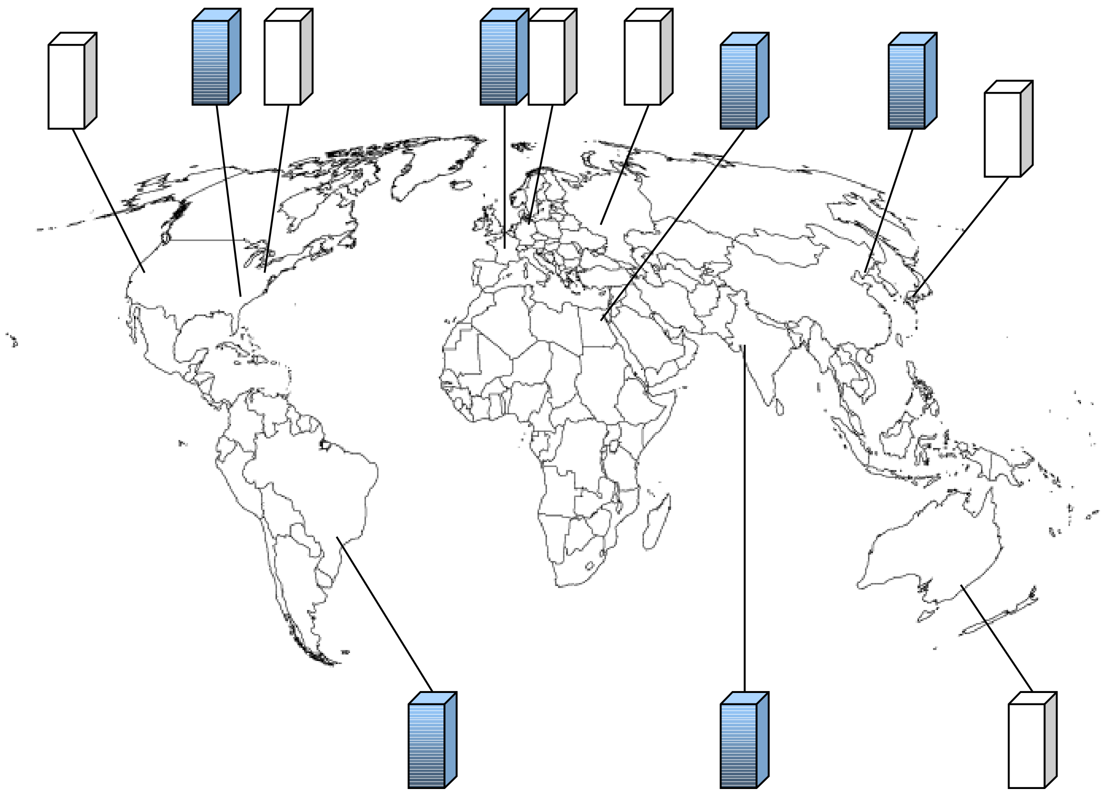


Figure 8: a world of constellations

When a user requests a routing page, we want to link her or him to one among the closest PageBoxes. As there is no public table IP subnet => geographical location we must avoid spending more time to find better information than we possibly can spare on requests once we have that information.

We chose countries as geographical atoms. We found tough to identify relevant sub-entities, states, prefectures, lands or whatever. We will rather focus on creating different levels of regions.

So when the Top Level Domain (TLD) is a country code, it provides the needed geographical information. The problem is that on about 32 millions of domains, 22 are .com and four millions .net.

We know only two related pieces of information about the requestor, its IP address and its host name. They are related as DNS converts the host name in IP address but they are managed by different authorities.

Most of the time, it works in that way:

1. An entity (an ISP or ASP, most of the time) buys a range of IP addresses
2. The entity creates an infrastructure with at least two DNS servers that handle Lower Level Domains and hosts on these domains

3. A second entity buys a service such as hosting to the first entity.
4. The second entity buys a domain and provides the IP addresses of the first entity DNS servers to the domain authority

As a consequence:
?? An IP address range hosts often a large number of domains
?? The IP address range owner location is meaningful for routing but the domain owner location is not

We can get needed information about IP addresses using the whois server of three registries:
1. RIPE (http://www.ripe.net/) that manages IP addresses for Europa , Middle East, and parts of Asia and Africa
2. APNIC (http://www.apnic.net/) that manages IP addresses for Asia
3. ARIN (http://www.arin.net/) that manages IP addresses for America and the rest of the world

Mapper first looks at the TLD and uses if it is a country code. If it is not, it queries registries for requestor IP address. As it is a time-consuming and expensive process, it caches address range.

6.2.1 Caching
Mapper maintains a cache range_of_IP_addresses | candidate_PageBoxes.
Therefore all users in the same range of IP addresses are routed to the same PageBox set.

Candidate PageBoxes include broken PageBoxes.
Mapper maintains another PageBox table where it maintains PageBox status. It sends periodically and asynchronously “HTTP pings” to PageBoxes to update the table.

So even in case of cache hit, it performs two steps:
?? To remove broken PageBoxes from the candidate list
?? To select one of them using a round-robbin algorithm

6.2.2 Search method
Mapper tries to find PageBoxes:
1. In the same domain
2. With the same TLD if it is a country code (not a generic TLD). If there is no candidate PageBox in the country, Mapper tries to find PageBoxes in the same region.
3. Otherwise Mapper queries ARIN, RIPE or APNIC for the requestor IP address. It gets a country code and a range of addresses. Then it looks for PageBoxes with the same country code and if there is no candidate in the country for PageBoxes in the same region.

6.2.3 Algorithm

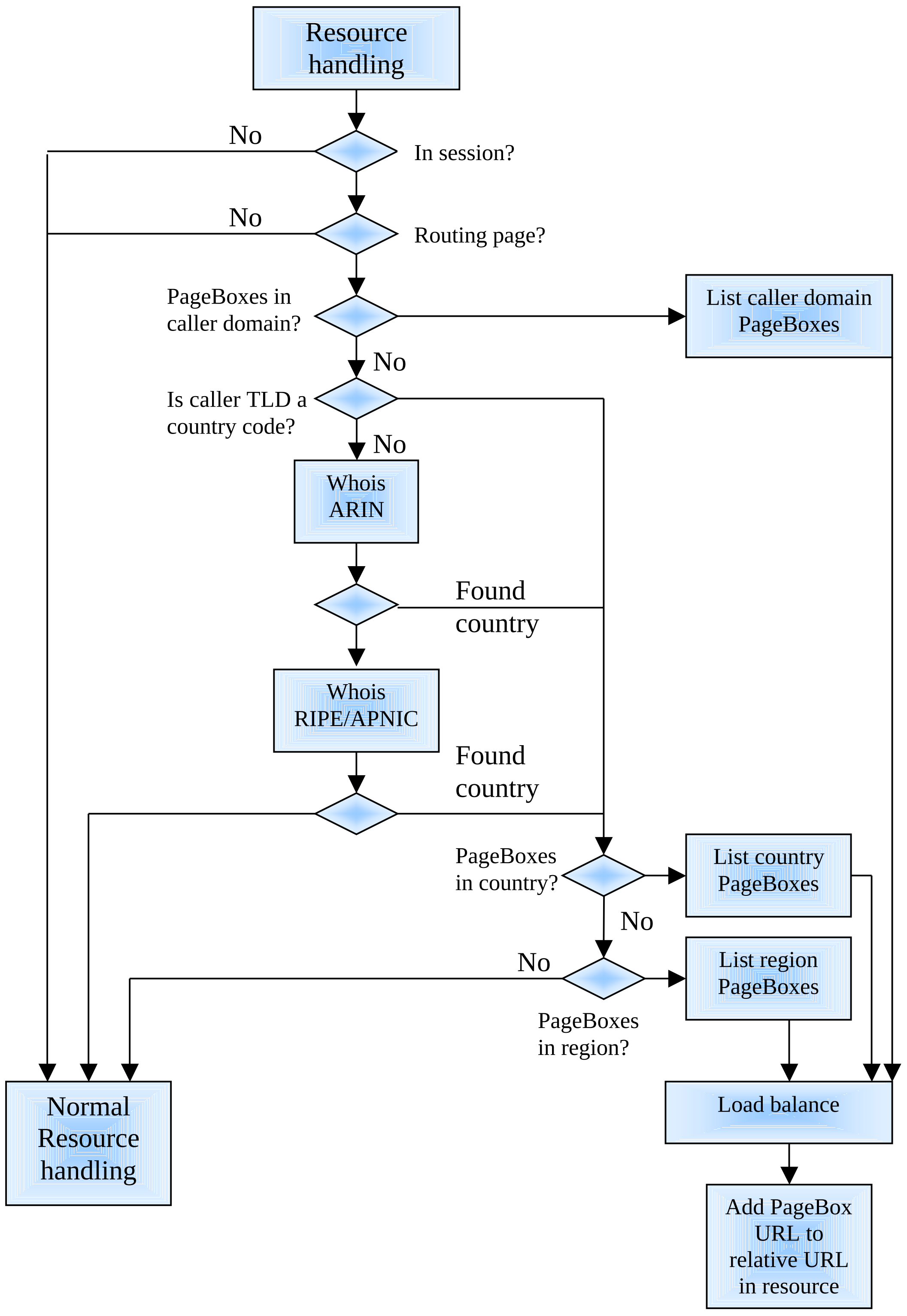


Figure 9: algorithm

6.2.4 Writing a mapper

A mapper class must be named Mapper and implement the MapperIntf interface:

```
interface MapperIntf {
  String getPageBox(String url);
  void checkAll(int counterForPing);
  boolean check(String subscriber);
  String getMessage();
  String getInfo();
  String getDomain(String host);
}
```

getPageBox is the core method that returns a PageBox URL, given the domain name of the requestor. It is invoked in routing page handling.

checkAll is the method responsible to “ping” PageBoxes. It is invoked by a scanner thread. It invokes the check method described below to perform the actual ping.

check is responsible to check if the PageBox whose URL is passed is up. It returns true if the PageBox is up and false otherwise.

getDomain extracts the domain name from an host name.

getMessage returns null or an error message and getInfo returns data about Mapper state. These data are used only for logging and by the monitoring URL.

Mapper must have a constructor:

```
Mapper(String publisher, JSPhandler handler)
```

publisher is the URL of a repository. Mapper must query the publisher for the list of subscribing PageBoxes.
handler is a JSPhandler. It provides a log object for tracing and a cache object.

You can also write your own cache. It must be named Cache and implement the CacheIntf interface:

```
interface CacheIntf {
  String report();
  void register(long address1, long address2, String code);
  String retrieve(long address);
  void remove(long address);
  void clear(int nbdays);
  boolean save();
}
```

report returns a report string about cache content. It should be called by Mapper.getInfo.
register records an address range in the cache with its country code.
Given an address, retrieve returns a country code.
register and retrieve are called by Mapper.
remove removes the address range that includes the address parameter from the cache.
clear asks for the removal of cache entries older than nbdays.
save asks to persist the cache.

CacheIntf doesn't include a restore method.
Cache should restore its state in its constructor whose signature is:

```
Cache(String cacheLocation)
```

We recommend starting from the existing mapper to write a new mapper and

1. To query PublisherServer to get a subscriber list
2. To use the PageBox ping to check if a PageBox.

Most of your changes should be changing tables of regions.

6.3 Use on Intranet

Mapper is designed primarily for Internet and Extranet environments.
It can also been used on Intranet without whois server. In that case, it relies only upon the top-level domain that it assumes to be a country code.
Note that if your host names don't include dots, it uses the full host name as country code.

From a Mapper point of view a country code is only a way to identify a set of computers and a region a way to identify a set of countries. So an internal top-level domain and even a host name are valid countries you can gather in regions as you want. Below we show a definition without network at all:

```
static final Region Test = new Region("Test", new String[] {"LOCALHOST",
   "AGRANDEM"});
```

In that case we declared a Test region with two countries: LOCALHOST because PageBox was started on localhost:8080 and AGRANDEM because the requestor host name was agrandem.