4

Case study

The VP'63 System

VP´63 System

Converting a centralized mainframe-based Architecture
into a
Distributed Client-Server Architecture

Some History

- An imaginary traditional Portuguese wine company owns an obsolete information system.
- The company management has devised an ambitious strategy for enhancement of the system in support of current and emerging business, and has contracted a team to develop an architectural project for that development.

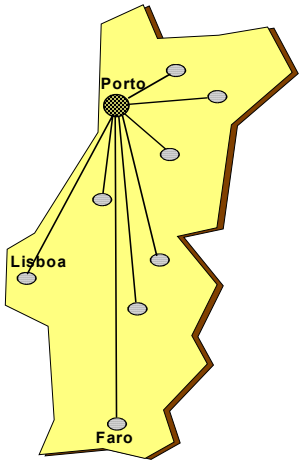
The Strategy (0)

- The strategy makers drafted the following objectives:
  - Seamless business information support system
  - Coherent document management support system
  - One-PC-per-employee strategy
  - Improved automation of the wine processing facilities
  - Integrated industrial management support system
  - On-line transactions in two facets: a Web portal and a virtual enterprise network

Point Zero

- The company has several vineyards in the country, with local offices and processing plants in some of them.
- The central offices are in Porto, the capital of the famous Port Wine.
- The information system, like many others of the earlier generations, is mainframe-based, centralized, without Internet access.
- Remote facilities access it through remote login, via virtual terminals on PCs, connected to the mainframe through leased lines

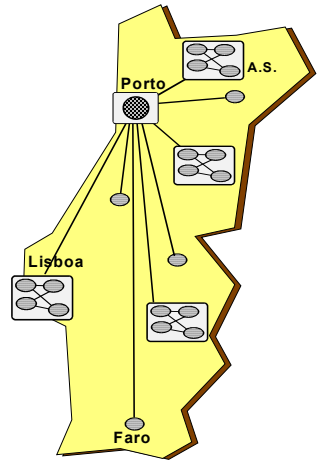
Centralized Mainframe (rlogin)



Point Zero dot One

- Some of the larger offices in the processing plants, have augmented their computing needs
- Mid-range servers with local databases and hosting local support services, were installed
- This evolution created a potential for inconsistency with the mainframe system
- That is, they became semi-autonomous subsystems or islands

Autonomous Subsystems/Islands (ftp)



Strategy for System Evolution

- maintaining centralized business control, whilst allowing the deployment of distributed services;
- achieving openess, through the use of COTS
  - (e.g., Linux and WNT), protocols (e.g., TCP/IP, HTTP), and infrastructures (e.g., Internet);
- distributing processing activities, for modularity in face of fast changing business configurations
- distribution of data repositories for information and resource sharing;
- enhancement with proprietary middleware when applicable;
- distribution should have in mind availability and performance enhancement, to be addressed in later stages.

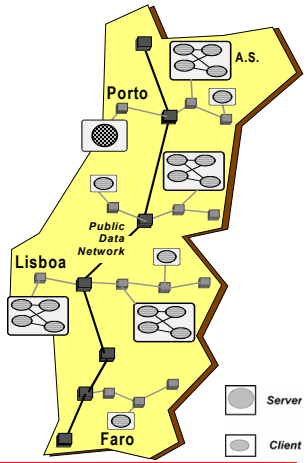
Distributed Computing Approaches

- What should be the evolution in terms of networking infrastructure?
  - migrate to the Internet
  - all islands fitted with routers connected to an ISP
  - PCs hosting remote terminals converted to client units, connected to the Internet

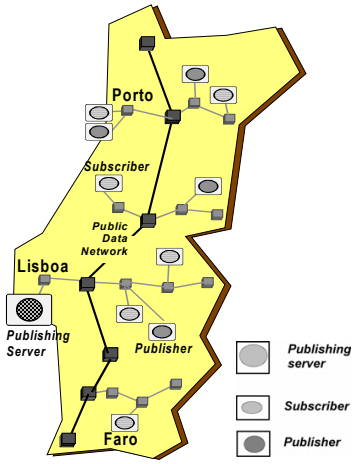
Distributed Computing Approaches

- What are the adequate paradigms in what concerns the organization of distributed activities and services in such a company?
  - *Client-server*, enabling: access to central services by remote facilities; access to services in the autonomous subsystems by the local clients; transactional access of Web clients, both internal (employees) and external (e-commerce clients).
  - *Publisher-subscriber*, enabling: event-based handling of generic management information in a content-sensitive manner; event-based handling of time sensitive production information, for seamless integration between the production management and the general management systems.
- How should cli-serv and pub-sub subsys be set up?

Client-Server



Publisher-Subscriber



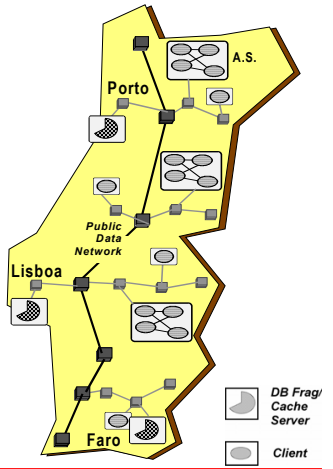
Distribution of Data Repositories
(problems)

- peak situations will become frequent when the main database in Porto will be overloaded;
- when the main database server in Porto or the connection to it fail, the whole enterprise operation stops;
- situations of network partitioning in long-haul connections from distant facilities are more probable;
- there is a potential for conflicting operations over the day between different autonomous islands, and between the latter and remote clients.

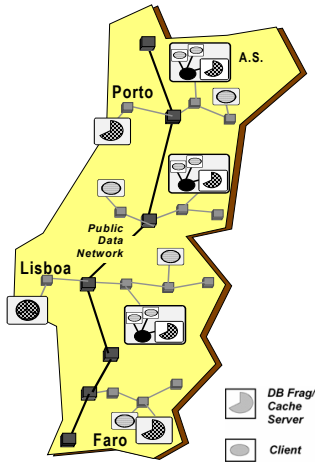
Distribution of Data Repositories
(solutions)

- How can the performance, availability and consistency problems created by a centralized database and ad-hoc caches be solved?
  - Distribution of the information repository, by means of a distributed database management system
  - Criteria for fragmentation should pay attention to data importance, functionality and locality
  - Criteria for distribution should match the criteria for fragmentation, placing fragments with local information in the relevant areas
  - the persistent information repositories of the islands should be redefined as fragments of the main database

Fragmentation of Central DB



Consolidation of the Island's DB Fragments



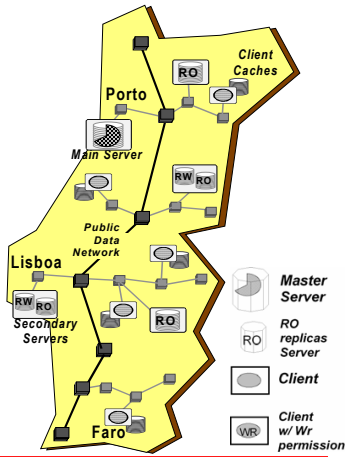
Distributed File System Access

- What kind of distributed file service is adequate for a geographically large-scale setting such as VP'63?
  - Large-scale distributed file system (DFS) with distributed read-write (RW) and read-only (RO) files or volumes
  - Upload-download type, with server files cached in local client disks, overcoming delay and instability of WAN communication
  - Transactional file access to both RO and RW volumes, for file-based information dissemination/archival, combined with event-based publisher-subscriber

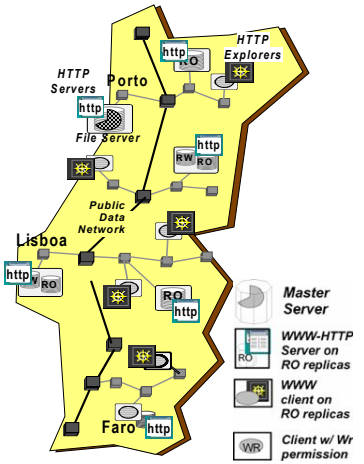
Distributed File System Access

- How to improve large/scale DFS operation?
  - Replicated RO volumes support long-term publishing. Shared RW volumes support moderately frequent single- or multiple-writer updates
  - Normal users access through the Web: provide 3-tier Web-based access to the large-scale DFS.
  - File system servers fuel DFS client's caches in the company infrastructure, as the second level of the hierarchy.
  - HTTP servers run on both servers and caches, serving pages to the third level of the hierarchy, the HTTP client browsers

Distributed File System



Web-based File Access

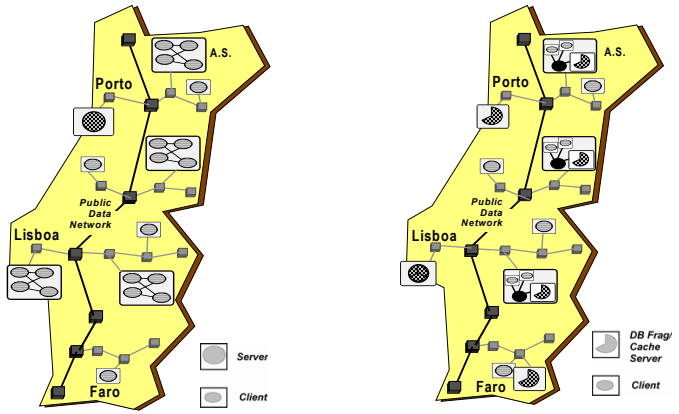


VP´63 System

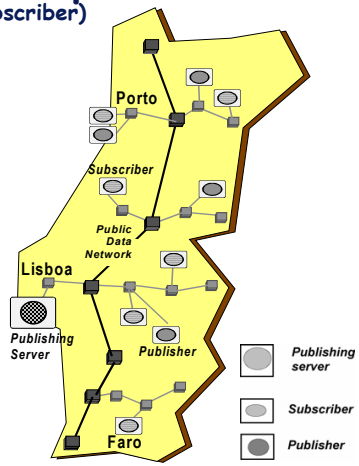
Making the distributed system

Fault-Tolerant

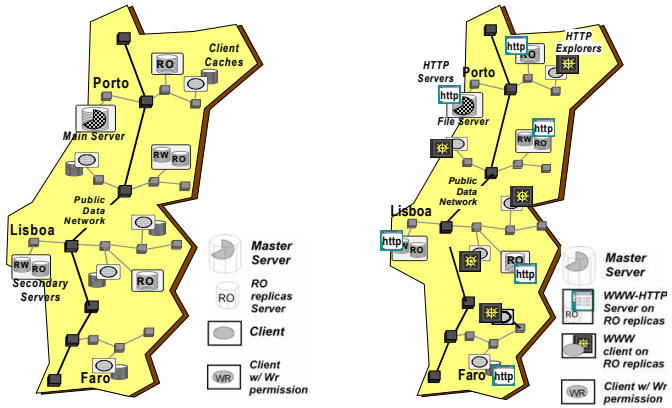
VP'63 re-cap
(distributed C/S frag DB version)



VP'63 re-cap
(Publisher-Subscriber)

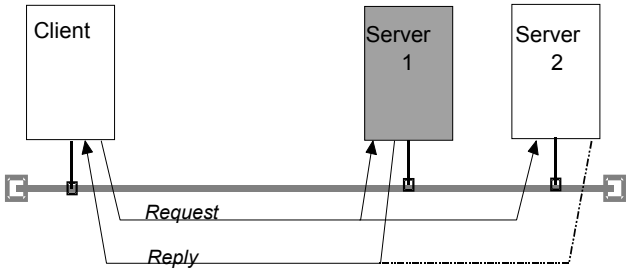


VP'63 re-cap
(DFS/WWW dissemination subsystem)

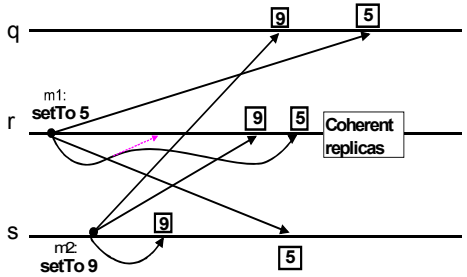


recapitulating some useful principles

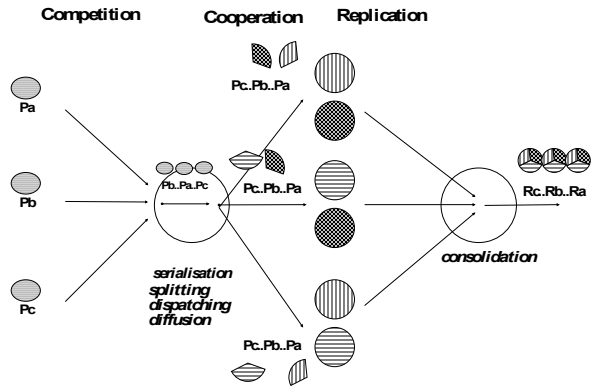
Replicated C/S principle



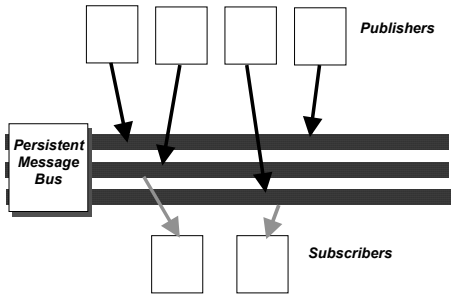
Atomic broadcast principle



Combined replication and fragmentation



Publisher-subscriber principle



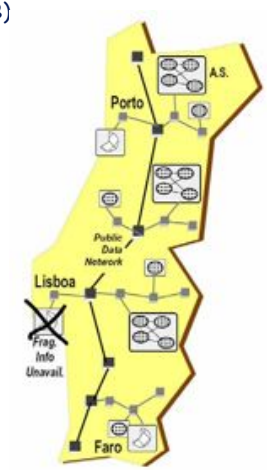
Initial situation

- The client-server front-end to the database is already fragmented. This is a first step at independence of failure local transactions will not be affected by the failure of servers of the other fragments.
- However this is not enough to guarantee overall availability, since operation related with that fragment stalls.
- On the other hand, the publisher-subscriber infrastructure is based on a single-location server (Lisboa) which, once down, stalls the whole service.
- This is a severe availability impairment.

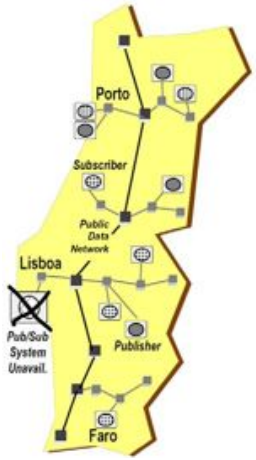
Initial situation (2)

- The 3-tier DFS-Web architecture has already a few aspects enhancing availability:
  - RO file or volume replicas are accessible by any client
  - upon failure of the local RO copy, the client can fetch a remote one.
  - This may eventually be extended to replication of RW les or volumes, with a DFS that support this feature.

Failure Scenarios
(Fragmented DB)



Failure Scenarios
(Pub-Sub System)



F/T Client-Server Database (1)

- What can be done to improve the availability of the database server?
  - go for a replicated distributed database server.
  - all fragments are now replicated in other sites. E.g., level of replication of n = 2 normally yields availabilities in excess of two nines
  - since the database is fragmented by enough sites it is not necessary to allocate extra machines to achieve fault tolerance.
  - modular fault tolerance is used, cross-allocating fragment replicas to sites containing other fragments
  - tests showed that risk of network partitioning is non-negligible in the links of the main inter-city connections
  - this can make the replica pairs diverge.

F/T Client-Server Database (2)

- What can be done to address partitioning of the database replicas?
  - If fragments are replicated at least in triplets, the primary-partition consistency criterion can be used, preventing divergence by allowing progress only in a partition with the majority of replicas.

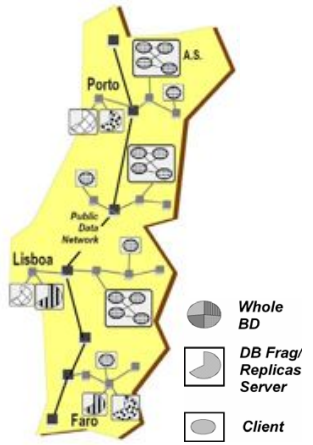
F/T Client-Server Database (3)

- Considering that the DBMS used allowed modular fragmentation and replication, will there be situations where it makes sense to allocate different redundancy levels to different fragments?
  - it was decided to group crucial data in a main fragment (MF) and replicate it in the main site and all islands, increasing the level of redundancy of this data
  - this configuration study will later be extended to other data.

F/T Data Dissemination

- What can be done to improve the availability of the pub-sub server?
  - With a simplex server, whole dissemination system stops when the server is down, and data may be lost
  - unless there is a transactional interface between publishers and server, and the latter has persistent storage.
  - possible solution: set up a replicated publisher-subscriber server
  - has additional benefit of improving performance of the scheme since subscribers get in general nearer the information bus materialized by the publishing server.
  - ex.: data are published through all replicas, subscribers get data from one of them
  - load of dissemination to the subscribers can be distributed by the publishing servers.

Fault Tolerance
(Client-Server Database)



Fault Tolerance
(Publisher-Subscriber)