ACM SIGOPS European Workshop 1988
Autonomy or Interdependence in Distributed Systems?
CERN Experience

Tim Berners-Lee

Data Handling (DD) division, Online group
CERN
1211 Geneva 23
Switzerland
Tel: +41 (22) 833755

March 1988

1. Background

At CERN (the European Particle Physics Laboratory) large numbers of minicomputers, workstations, and microprocessors embedded in complex FASTBUS and VME based systems are used for data acquisition and the control of particle physics experiments. The design of distributed software to run on many different types of processor and operating systems has provoked the development of a an infrastructure based on Remote Procedure Call, and many related tools and applications. It has also highlighted practical constraints and requirements in a real-life heterogeneous systems.

2. Autonomy or Interdependence?

There are two interesting angles to this question. The first is about the distribution of basic functions, and the second is about the distribution of the management (control) of those functions. (I use the term "management" to avoid confusion with process control functions).

3. Why a distributed system?

At a recent workshop on distributed systems one researcher was heard to ask why distributed systems are needed at all, as opposed to centralised ones. The answers to this question (there are several answers at CERN) are clues to the autonomy/interdependence questions.

Reasons for the distribution of systems at CERN include

1. The throughput. Experiments have to cope with physics "event" data of the order of 200kB (after compression), at rates of kilohertz. This data rate can only be filtered and treated by using a high degree of parallelism, and distributed management. Typically, of the order of 100 M68020 based processors will be distributed around the experiment. In this requirement, CERN may be unusual among users of distributed systems. However, many distributed systems have been designed to share computing load (The Apollo NFS system is an example).

2. Resource sharing. The distributed processors are normally dedicated to particular parts of the experiment, but they must share common resources such as databases and error reporting facilities. This seems to be the reason behind most early distributed systems (for example, the Sun NFS system based on XDR).

3. Geography. Experiments associated with the LEP accelerator are necessarily situated up to 150 metres below ground level, in an environment which is unsafe for people while the experiment is running. This makes the distribution of processing power essential, as the large computers, and data storage facilities are above ground.

These are all primary reasons for a distributed system, and determine the distribution of basic functions. A fourth reason is also important, particularly when considering management functions.

4. Distributed design. The total complex of accelerators and experiments constitute a very large project. To coordinate the whole as one enormous project would be an undertaking comparable with the NASA space program. The teams contributing to different parts of it come

from institutes all over the world, often having designed parts of the experiments at their home institutes. When they arrive at CERN, these are integrated with existing systems. There is no top-down design of the whole system. In terms of traditional software engineering principles, this would have been regarded as a disaster, but in practice there is a limit to the size of project which can be designed from the top down.

The formation of the whole system therefore consists of the amalgamation of existing systems, with a continuous negotiation between project teams about the interfaces between their systems. This is the practicality at CERN, and must also be the practicality when one attempts to combine distributed systems from more than one vendor.

4. Open System Interconnection

Put another way, the ability for previously independent systems to be linked is simply the "Open Systems" requirement (as in OSI). This was recognised many years ago as a general requirement of communications systems, and distributed systems are no exception: it is just that they are coming of age more slowly.

The introductory note to this workshop suggested two models of management, local management and global management. The "open system" constraint rules out global management, insisting that a system of any size has its own measure of management. Where complete autonomy is impractical, a flexible system must exist, of which complete autonomy and global management become special cases. This can be achieved by the delegation of management between systems.

Consider the problem of finding a provider for a needed service or resource. A software module charged with resolving this need may take the following actions:

1. Produce results from its own local tables, the results of negotiation with other known systems (autonomy);

2. Refer the problem blindly to a single global manager (global management);

3. Refer the problem to a relevant manager who has accepted responsibility for certain ranges of cases.

The flexibility needed requires approaches 1 and 3, giving a blend of autonomy and interdependence which can allow a large system of arbitrary topology.

5. System Topology

Action 3 above may involve different requests being forwarded to different managers, making a multiply-connected, and not hierarchical, topology.

In practice, the topology of a large system will follow two things: the topology imposed by the problem it is designed to solve, and the topology of the organisation which designed it. If it is not naturally hierarchical, it should not be forced into hierarchical form by constraints of the distributed system software.

We conclude with three design rules:

1. The local manager has priority in the need for a service is being resolved;

2. The need may be resolved locally, or passed on (recursively) to another authority who accepts responsibility for it;

3. To impose a hierarchical structure on the paths of interconnection is overconstraining.

6. Summary

Autonomous systems and highly interdependent ones should be special cases of a general case, based on the delegation of responsibility for coordination to local subsystem managers.

[Current RPC systems at CERN use autonomous management, with local tables of server addresses updated remotely. A distributed configuration manager along the lines indicated above is at the specification stage].