M.I.T. Laboratory for Computer Science October 21, 1976

Computer Systems Research Division Request for Comments No. 128

DISCUSSIONS ON "DISTRIBUTED SYSTEM" RESEARCH

by J. H. Saltzer

Two recent staff meetings have been devoted to discussion of the current research proposal to ARPA concerning "distributed systems". Several interesting questions have come up, so this memo is intended to capture those questions, some of the discussion, and some further thoughts of my own, and to stimulate further thoughts.

What kind of "distributed system" is the right vehicle for research?

Much of the initial (and later) discussion involved developing an image of what kind of distributed systems are envisioned by the proposal. Two interpretations seem possible.

1) A system that looks to its users and behaves at its interfaces as though it were a monolithic central system, but is behind the scenes actually engineered from physically or geographically distributed pieces: (A "distributed Multics".)
2) A system whose external image is "this computer is self-contained, but is prepared to be a cooperating member of a set of similarly cooperating computers".

The second image is closer to the target, for several reasons:

1) If done properly and desired for some particular application, the extreme of cooperation would allow the second system to include the first as an example. The reverse is apparently not true.
2) With less extreme forms of cooperation (ranging down to none at all) the important real world administrator's statement of "I want a local system so that our work is completely under our control" can be addressed. Here is where the high impact on the future potentially lies.

Thus, we should be thinking more in terms of gluing together systems from smaller systems, and less in terms of tearing big systems apart into physically separable modules. One way to emphasize this point might be to rename the project "Research in Integrated Systems". (That renaming would be premature, but suggestions for a project name that better captures out goals are welcome.)

Put another way, it appears that lower computer prices are making feasible system acquisition strategies that people always wanted, but could rarely afford: dedication of computers to one clearly identifiable function. This dedication makes it easier for everyone involved to understand what the computer is for, its relative importance, and their relation to it--all administrative and management goals that are hard to achieve when one machine is used for diverse purposes. So the problem is to allow for any desired level of (possibly later) coherent integration of systems that have a natural excuse for a separate existence.

Then are we integrating arbitrary heterogeneous systems?

Not yet. We should restrict our attention to this question: can we plan the design of a new system so that it is prepared for any level of coherent coordination with other similarly prepared (but possibly of different internal design) systems of the future? There is an alternative high impact present day problem, namely creating coherence across computer systems already in someone's possession. That problem seems to consist of the "coherence across prepared systems" problem plus a whole lot of engineering constraints that must becloud any fundamental issues. Although someone might explore some of these engineering issues, I think that our main thrust must be for now on the fundamental issue of attaining coherence and integration in a favorable environment. Once that area is understood, it may be clearer how extensions to older systems could be feasible.

Didn't NBC News report that the ARPANET already solves this problem?

The ARPANET provides a first, important cut at providing a form of coherence, in that it defines a standard way for one system to get a message to another, and also several standard ways to use those messages for simple functions such as logging in. Some experimental protocols on the ARPANET

(principally RSEXEC and NSW) have explored coherence at a deeper level, namely in the file cataloging systems. That is roughly our starting point. Some possible examples of deeper problems in coherence are: how can one embed in one file a cross-reference to something in a file stored elsewhere? What should a program do when it encounters such a reference? Can the "other" file actually be present in multiple copies at different places, for reliability? If I only suspect the existence of a file I need, how can I interactively search for it? Can I obtain information from a distant file without knowing anything about its internal format, and with minimum bother to its owner?

Probably the main mechanical difference between our strategy and the ARPANET strategy is that we should assume a deeper level of change to be acceptable in preparing a system for cooperation than was practical in developing the ARPANET. New addressing architectures, new "file systems", and more "object-oriented" views all seem appropriate. Looking into higher levels of the system may be necessary.

Does this mean building a system?

At some stage, almost certainly yes. For the moment, though, discussions of proper designs should be sufficient to get started. One of the main reasons for the current activity of building a local network for the laboratory is to provide an environment in which an actual system implementation could be carried out. The protocol design for the LCS local network is in a sense our very first (simple) example of a system for coherently integrating distinct systems.

In designing a new system, to what extent should the Multics design be a basis or a constraint?

Given the history of the group, it seems unlikely that any new system design will be uninfluenced by Multics. Two things are crucial here:

1) Multics addressed many difficult issues and provided solutions not yet available in any other form; it demonstrated considerations in system design that have not yet been explored elsewhere. We must take care to build on this base of knowledge where appropriate, and not waste time rediscovering it.

2) There are now new goals, and new ideas in system organization, and the old strategies must not be permitted to become a constraint on what is done in the future.

Perhaps a way to address this problem is to introduce a kind of intellectual "zero-base-budgeting" to the design. That is, for a concept or design idea to be acceptable, it should be accompanied by an argument based on fundamental requirements. An argument based on "but this is the way it was done in Multics" is insufficient. On the other hand, Multics provides a kind of counter-example and test case. If a design concept seems to lead to a function inferior to that already available in Multics, a very careful understanding of the situation should be demanded.

What kind of applications should be the assumed customer for the kind of system we are designing?

1) The application has a strong natural reason to be decentralized. That is, the administrative and managerial pressures described earlier are present.
2) The application has a natural need for coherence beyond that provided, say, by ARPANET TELNET, FTP, and mail/message service protocols. Probably naming and cross references among files/objects is required.
3) The application should be dynamic enough to raise interesting problems. The dynamics should be in structure, rather than in data value. For example, an airline reservations system deals with relatively static structures (the flight lists) but the data values are very dynamic. On the other hand an airline operations system might require dynamic structures.* Certainly a military command-and-control system has this kind of dynamics, but there are practical problems in learning enough about that kind of application for it to be of any help.

---

* An airline operations system is the system that helps the vice president of operations cope with the following problem: you have 30 planes on the ground at Chicago, and 50 planes in the air headed there. They just closed the Chicago airport because of snow. What do you do now?

It may turn out that we should take on some specific applications in our work. In that case, a considerable problem surrounds not being overwhelmed by details of or research in the application itself. Our interest (and ability) is in the underlying system substrate that can be used for a class of applications, and it is important that our primary intellectual energy be focused there.