Memorandum For Members and Affiliates of the Intergalactic Computer Network

December 11, 2001 by J.C.R. Licklider

This memo sent from J.C.R. Licklider to his colleagues in 1963 explores the early challenges presented in trying to establish a time-sharing network of computers with the software of the era–ultimately, this vision would lead to ARPANet, the precursor of the Internet in use today. Will the future iterations lead to an Intergalactic Computer Network?

Originally distributed as a memorandum April 23, 1963. Published on KurzweilAI.net December 11, 2001.

ADVANCED RESEARCH PROJECTS AGENCY

Washington 25, D.C. April 23, 1963

MEMORANDUM FOR: Members and Affiliates of the Intergalactic Computer Network

FROM: J. C. R. Licklider

SUBJECT: Topics for Discussion at the Forthcoming Meeting

First, I apologize humbly for having to postpone the meeting scheduled for 3 May 1963 in Palo Alto. The ARPA Command & Control Research office has just been assigned a new task that must be activated immediately, and I must devote the whole of the coming week to it. The priority is externally enforced. I am extremely sorry to inconvenience those of you who have made plans for May 3rd. Inasmuch as I shall be in Cambridge the rest of this week, I am asking my colleagues here to re-schedule the meeting, with May 10th, Palo Alto, as target time and place.

The need for the meeting and the purpose of the meeting are things that I feel intuitively, not things that I perceive in clear structure. I am afraid that that fact will be too evident in the following paragraphs. Nevertheless, I shall try to set forth some background material and some thoughts about possible interactions among the various activities in the overall enterprise for which, as you may have detected in the above subject, I am at a loss for a name.

In the first place, it is evident that we have among us a collection of individual (personal and/or organizational) aspirations, efforts, activities, and projects. These have in common, I think, the characteristics that they are in some way connected with advancement of the art or technology of information processing, the advancement of intellectual capability (man, man-machine, or machine), and the approach to a theory of science. The individual parts are, at least to some extent, mutually interdependent. To make progress, each of the active research needs a software base and a hardware facility more complex and more extensive than he, himself, can create in reasonable time.

In pursuing the individual objectives, various members of the group will be preparing executive the monitoring routines, languages amd [sic.] compilers, debugging systems and documentation schemes, and substantive computer programs of more or less general usefulness. One of the purposes of the meeting–perhaps the main purpose–is to explore the possibilities for mutual advantage in these activities–to determine who is dependent upon whom for what and who may achieve a bonus benefit from which activities of what other members of the group. It will be necessary to take into account the costs as well as the values, of course. Nevertheless, it seems to me that it is much more likely to be advantageous than disadvantageous for each to see the others' tentative plans before the plans are entirely crystalized. I do not mean to argue that everyone

should abide by some rigid system of rules and constraints that might maximize, for example, program interchangeability.

But, I do think that we should see the main parts of the several projected efforts, all on one blackboard, so that it will be more evident than it would otherwise be, where network-wide conventions would be helpful and where individual concessions to group advantage would be most important.

It is difficult to determine, of course, what constitutes "group advantage." Even at the risk of confusing my own individual objectives (or ARPA's) with those of the "group," however, let me try to set forth some of the things that might be, in some sense, group or system or network desiderata.

There will be programming languages, debugging languages, time-sharing system control languages, computer-network languages, data-base (or file-storage-and-retrieval languages), and perhaps other languages as well. It may or may not be a good idea to oppose or to constrain lightly the proliferation of such. However, there seems to me to be little question that it is desireable to foster "transfer of training" among these languages. One way in which transfer can be facilitated is to follow group consensus in the making of the arbitrary and nearly-arbitrary decisions that arise in the design and implementation of languages. There would be little point, for example, in having a diversity of symbols, one for each individual or one for each center, to designate "contents of" or "type the contents of." It seems to me desirable to have as much homogenity as can reasonably be achieved in the set of sub-languages of a given language system–the system, for example, of programming, debugging, and time-sharing–control lanugages related to JOVIAL on the Q-32, or the system related to Algol (if such were developed and turned out to be different from the JOVIAL set) for the Q-32 computer, or the set related to FORTRAN for a 7090 or a 7094.

Dictating the foregoing paragraph led me to see more clearly than I had seen it before that the problem of achieving homogeneity within a set of correlated languages is made difficult by the fact that there will be, at a given time, only one time-sharing system in operation on a given computer, whereas more than one programming language with its associated debugging language may be simultaneously in use. The time-sharing control language can be highly correlated only with one programming and debugging language pair. Insofar as syntax is concerned, therefore, it seems that it may be necessary to have a "preferred" language for each computer facility or system, and to have the time-sharing control language be consistent with the preferred. Insofar as semantics is concerned–or, at least, insofar as the association of particular symbols with particular control functions is concerned–I see that it would be possible, thought perhaps inconvenient, to provide for the use, by several different operators, of several different specific vocabularies. Anyway, there seems to me to be a problem, or a set of problems, in this area.

There is an analogous problem, and probably a more difficult one, in the matter of language for the control of a network of computers. Consider the situation in which several different centers are netted together, each center being highly individualistic and having its own special language and its own special way of doing things. Is it not desirable, or even necessary for all the centers to agree upon some language or, at least, upon some conventions for asking such questions as "What language do you speak?" At this extreme, the problem is essentially the one discussed by science fiction writers: "how do you get communications started among totally uncorrelated "sapient" beings?" But, I should not like to make an extreme assumption about the uncorellatedness. (I am willing to make an extreme assumption about the sapience.) The more practical set of questions is: Is the network control language the same thing as the time-sharing control language? (If so, the implication is that there is a common time-sharing control language.) Is the network control language different from the time-sharing control language, and is the network-control language common to the several netted facilities? Is there no such thing as a network-control language? (Does one, for example, simply control his own computer in such a way as to connect it into whatever part of the already-operating net he likes, and then shift over to an appropriate mode?)

In the foregoing paragraphs, I seem to have lept into the middle of complexity. Let me approach from a different starting point. Evidently, one or another member of this enterprise will be preparing a compiler, or compilers, for modifying existing programs that compile FORTAN [sic.], JOVIAL, ALGOL, LISP and IPL-V

(or V-l, or V-ll). If there is more than one of any one of the foregoing, or of any one of others that I do not foresee, then it seems worthwhile to examine the projected efforts for compatibility. Moreover, to me, at least, it seems desireable to examine the projected efforts to see what their particular features are, and to see whether there is any point in defining a collection of desireable features and trying to get them all into one language and one system of compilers. I am impressed by the argument that list-structure features are important as potential elements of ALGOL or JOVIAL, that we should think in terms of incorporating list-structure features into existing languages quite as much as in terms of constructing languages around list-structures.

It will possibly turn out, I realize, that only on rare occasions do most or all of the computers in the overall system operate together in an integrated network. It seems to me to be interesting and important, nevertheless, to develop a capability for integrated network operation. If such a network as I envisage nebulously could be brought into operation, we would have at least four large computers, perhaps six or eight small computers, and a great assortment of disc files and magnetic tape units–not to mention the remote consoles and teletype stations–all churning away. It seems easiest to approach this matter from the individual user's point of view–to see what he would like to have, what he might like to do, and then to try to figure out how to make a system within which his requirements can be met. Among the things I see that a user might want to have, or to do, are the following:

(Let me suppose that I am sitting at a console that includes a cathode-ray-tube display, light-pen, and a typewriter.) I want to retrieve a set of experimental data that is on a tape called Listening Test. The data are called "experiment 3." These data are basically percent- ages for various signal-to-noise ratios. There are many such empirical functions. The experiment had a matrix design, with several listeners, several modes of presentation, several signal frequencies, and several durations. I want, first, to fit some "theoretical" curves to the measured data. I want to do this in a preliminary way to find out what basic function I want to choose for the theoretical relation between precentage [sic.] and signal-to-noise ratio. On another tape, called "Curve Fitting," I have some routines that fit straight lines, power functions, and cumulative normal curves. But, I want to try some others, also. Let me try, at the beginning, the functions for which I have programs. The trouble is, I do not have a good grid-plotting program. I want to borrow one. Simple, rectangular coordinates will do, but I would like to specify how many divisions of each scale there should be and what the labels should be. I want to put that information in through my typewriter . Is there a suitable grid-plotting program anywhere in the system? Using prevailing network doctrine, I interrogate first the local facility, and then other centers. Let us suppose that I am working at SDC, and that I find a program that looks suitable on a disc file in Berkeley. My programs were written in JOVIAL.

The programs I have located throught the system were written in FORTRAN. I would like to bring them in as relocatable binary programs and, using them as subroutines, from my curve-fitting programs, either at "bring-in time" or at "run-time."

Supposing that I am able to accomplish the steps just described, let us proceed. I find that straight lines, cubics, quintics, etc., do not provide good fits to the data. The best fits look bad when I view them on the oscilloscope.

The fits of the measured data to the cumulative normal curve are not prohibitively bad. I am more interested in finding a basic function that I can control appropriately with a few perimeters than I am in making contact with any particular theory about the detection process, so I want to find out merely whether anyone in the system has any curve- fitting programs that will accept functions supplied by the user or that happen to have built-in functions roughly like the cumulative normal curve, but assymmetrical. Let us suppose that I interrogate the various files, or perhaps interrogate a master-integrated, network file, and find out that no such programs exist. I decide, therefore, to go along with the normal curve.

At this point, I have to do some programming. I want to hold on to my data, to the programs for normal curve fitting, and to display programs that I borrowed. What I want to do is to fit cumulative normal curves to my various sub-sets of data constraining the mean and the variance to change slowly as I proceed along any of the ordinal or ratio- scale dimensions of my experiment, and permitting slighly different sets of perimeters for the various subjects. So, what I want to do next is to create a kind of master program to set perimeter values for

the curve-fitting routines, and to display both the graphical fits and the numerical measures of goodness to fit as, with light-pen and graphics of perimeters versus independent variables on the oscilliscope screen, I set up and try out various (to me) reasonable configurations. Let us say that I try to program repeatedly on my actual data, with the subordinate programs already mentioned, until I get the thing to work.

Let us suppose that I finally do succeed, that I get some reasonable results, photograph the graphs showing both the empirical data and the "theoretical" curves, and retain for future use the new programs. I want to make a system of the whole set of programs and store it away under the name "Constrained-perimeter Normal-curve-fitting System."

But, then suppose that my intuitively natural way of naming the system is at odds with the general guidelines of the network for naming programs. I would like to have this variance from convention called to my attention, for I am a conscientious "organization man" when it comes to matters of program libraries and public files of useful data.

In the foregoing, I must have exercised several network features. I engaged in information retrieval through some kind of system that looked for programs to meet certain requirements I had in mind. Presumably, this was a system based upon descriptors, or reasonable facsimiles thereof, and not in the near future, upon computer appreciation of natural language. However, it would be pleasant to use some of the capabilities of avant-garde linguistics. In using the borrowed programs, I effected some linkages between my programs and the borrowed ones. Hopefully, I did this without much effort–hopefully, the linkages were set up–or the basis for making them was set up–when the programs were brought into the part of the stytem [sic.] that I was using. I did not borrow any data, but that was only because I was working on experimental data of my own. If I had been trying to test some kind of a theory, I would have wanted to borrow data as well as programs.

When the computer operated the programs for me, I suppose that the activity took place in the computer at SDC, which is where we have been assuming I was. However, I would just as soon leave that on the level of inference. With a sophisticated network-control system, I would not decide whether to send the data and have them worked on by programs somewhere else, or bring in programs and have them work on my data. I have no great objection to making that decision, for a while at any rate, but, in principle, it seems better for the computer, or the network, somehow, to do that. At the end of my work, I filed some things away, and tried to do it in such a way that they would be useful to others. That called into play, presumably, some kind of a convention-monitoring system that, in its early stages, must almost surely involve a human criterian as well as maching [sic.] processing.

The foregoing (unfortunately long) example is intended to be a kind of example of example. I would like to collect, or see someone collect, a considerable number of such examples, and to see what kind of software and hardware facilities they imply. I have it well in mind that one of the implications of a considerable number of such examples would be a very large random-access memory.

Now, to take still another approach to this whole matter, let me string-together a series of thoughts that are coming to mind. (I was interrupted at this point, and the discussion almost has to take a turn.) First, there is the question of "pure procedure." I understand that the new verion of JOVIAL is going to compile programs in "pure-procedure" style.

Will the other compilers at the other centers do likewise? Second, there is the question of the interpretation, at one center, of requests directed to it from another center. I visualize vaguely some kind of an interpretive system that would serve to translate the incoming language into commands or questions of the form in terms of which the interrogated center operates. Alternatively, of course, the translation could be done at the sending end. Still alternatively, the coordination could be so good that everybody spoke a common language and used a common set of formates. Third, there is the problem of protecting and updating public files. I do not want to use material from a file that is in the process of being changed by someone else. There may be, in our mutual activities, something approximately analogous to military security classification. If so, how will we handle it?

Next, there is the problem of incremental compiling. Am I correct in thinking that Perlis, with his "threaded lists," has that problem, and the related problem of com- pile-test-recompile, essentially solved?

Over on the hardware side, I am worried that the boundry- registered problem, or more generally the memory-protection problem, may be expensive to solve on the Q-32 and both difficult and expensive to solve on other machines, and I am worried that the problem of swapping or transferring information between core and secondary memory will be difficult and expensive on 7090s and 7094s–and I worry that time-sharing will not be much good without fast swaps or transfers. What are the best thoughts on these questions? In what state are our several or collective plans?

Implicit in the long example was the question of linking subroutines at run time. It is easy to do the calling, itself, through a simple directory, but it seems not to be so simple to handle system variables. Maybe it is simple in principle and perhaps I should say that it seems possibly infeasible to handle the linking of the system variables at run time through tables or simple addressing schemes.

It is necessary to bring this opus to a close because I have to go catch an airplane. I had intended to review ARPA's Command-and-Control interests in improved man-computer interaction, in time-sharing and in computer networks. I think, however, that you all understnad [sic.] the reasons for ARPA's basic interest in these matters, and I can, if need be, review them briefly at the meeting. The fact is, as I see it, that the military greatly needs solutions to many or most of the problems that will arise if we tried to make good use of the facilities that are coming into existence.

I am hoping that there will be, in our individual efforts, enought evident advantage in cooperative programming and operation to lead us to solve th problems and, thus, to bring into being the technology that the military needs. When problems arise clearly in the military context and seem not to appear in the research context, then ARPA can take steps to handle them on an ad hoc basis. As I say, however, hopefully, many of the problems will be essentially as important, in the research context as in the military context.

In conclusion, then, let me say again that I have the feeling we should discuss together at some length questions and problems in the set to which I have tried to point in the foregoing discussion. Perhaps I have not pointed to all the problems. Hopefully, the discussion may be a little less rambling than this effort that I am now completing.