A large computer program typically divided into many hundreds or even thousands of smaller units, whose logical connections define a network in a natural way. This network reflects the internal structure of the program, and defines the “information flow” within the program. We show that, (1) due to its growth in time this network displays a scale-free feature in that the probability of the number of links at a node obeys a power-law distribution, and (2) as a result of performance optimization of the program the network has a small-world structure. We believe that these features are generic for large computer programs. Our work extends the previous studies on growing networks, which have mostly been for physical networks, to the domain of computer software.
lying these programs. We provide strong evidence that the networks are scale-free and small worlds. While both the scale-free and small-world features have been demonstrated in many physical (or “hardware” type of) networks such as the Internet, the World Wide Web, and actor collaboration networks [1, 2, 24, 25, 26], our work demonstrates, for the first time, that these features also govern the network dynamics and topology in the software domain of computer science.

The programming language of choice for encoding large complex programs is C (and its offspring C++). In order to make a program manageable, the code is split into many small files. These files are of two kinds: source files and header files. The source files (usually with names terminating in “.c” or “.cpp”) contain the actual code, whereas the header files (with termination “.h”) have definitions of variables, constants, data structure and other information needed by the source files. A large program typically consists of thousands of source and header files. If a source file needs the information contained in a header file, that file is “included” in the source file with an “#include” clause. For example, if the source file “main.c” needs some data structure defined in “sys.h”, it contains a statement such as “#include <sys.h>”, whereby contents of “sys.h” are made accessible to “main.c”.

A network can now be defined from the set of source and header files, as follows. The nodes of the network are header files, and two nodes are defined to be connected if the corresponding header files are both included in the same source file. Connected header files are thus functionally related (they “work together” to help the source file in which they are both included do its job). By using a simple program that automatically scans every source file to see which header files each one of them includes, we generate the network corresponding to each of the four large programs aforementioned. We note that a few header files included in the source files belong to external libraries, and are not part of the program itself. When generating the networks, we ignore such files. Also, we only consider the largest connected component of the network, which includes over 90% of all nodes in all four cases.

We first present results concerning the scale-free feature of the computer-code networks. Let \( k \) be the variable that measures the number of links at different nodes in the network. For a network that contains a large number of nodes, \( k \) can be regarded as a random variable. Let \( P(k) \) be the probability distribution of \( k \). A scale-free network is characterized by the following algebraic scaling behavior in \( P(k) \):

\[
P(k) \sim k^{-\gamma},
\]

where \( \gamma \) is the scaling exponent. As pointed out in Refs. [1, 2, 24, 25], many real networks, such as the Internet, the World Wide Web, and the network of movie actors, appear to be scale-free with the exponent of value ranging from 2 to 3. The theoretical model proposed in Ref. [2] suggests the following two basic features in the network dynamics, which determine the algebraic scaling law: growth and preferential attachment. For growth, one can start with a small number \( m_0 \) of vertices and at every time step add a new vertex with \( m \) edges to the network, where \( m \leq m_0 \). For preferential growth, one can choose the probability that a new link is to be added to the \( k \)th node to be proportional to the number of links already existing in that node. The scaling law (1) can be derived from these two conditions [2]. Fig. 1 shows the scaling behavior of \( P(k) \) for three of the computer programs that we consider here, where (a-c) correspond to the Linux kernel, XFree86, and Mozilla, respectively. [The total number of nodes in the network associated with Gimp is too small to allow for the statistical quantity \( P(k) \) to be computed.] For large \( k \), a robust algebraic scaling behavior is present in all the three cases, where the scaling exponents are \( \gamma_{\text{Linux}} \approx 2.8 \), \( \gamma_{\text{XFree86}} \approx 2.9 \), and \( \gamma_{\text{Mozilla}} \approx 1.9 \). These results suggest that large computer programs can be regarded as scale-free, growing networks [27].

We next turn to the small-world feature of the large computer-program networks. For a given program, once the underlying network is built up, we can calculate the quantities that characterize their statistical properties; these are shown in Table I, for each program. We see that the average number of links per node \( \mu \) in all networks is much smaller than the total number of nodes \( N \), which means that the networks are sparse, a necessary condition for the notion of small-world network to be meaningful. The quantities of interest to us are the average shortest path \( L \), which is the average over all pairs of nodes of the number of links in the shortest path connecting the two nodes; and the clustering \( C \), which is the probability that two nodes \( a \) and \( b \) are connected, given that they are both connected to a common third node \( c \). If \( C \) is close to 0, the network is not locally structured; if \( C \) is close to 1, the network is highly clustered.

A random network with given \( N \) and \( \mu \) (with \( N \gg \mu \)) is characterized by having small values of \( L \) and \( C \). In particu-
TABLE I: Results for the networks corresponding to the four programs we have studied. \( N \) is the total number of nodes; \( \mu \) is the average number of links per node; \( C \) is the clustering coefficient, and \( C_{\text{rand}} \) is its value for an equivalent random network; \( L \) is the average shortest path, and \( L_{\text{rand}} \) is the same quantity for the corresponding random network.

\[
\begin{array}{|c|c|c|c|c|c|}
\hline
\text{program} & N & \mu & C & C_{\text{rand}} & L & L_{\text{rand}} \\
\hline
\text{Linux kernel} & 1448 & 41.4 & 0.88 & 0.03 & 2.11 & 1.93 \\
\text{Mozilla} & 3803 & 76.6 & 0.81 & 0.02 & 2.49 & 1.87 \\
\text{XFree86} & 1465 & 33.0 & 0.81 & 0.02 & 2.56 & 2.05 \\
\text{Gimp} & 403 & 43.9 & 0.83 & 0.11 & 2.28 & 1.56 \\
\hline
\end{array}
\]

ular, for \( N \rightarrow \infty \) and \( \mu \) fixed, the average shortest path in the largest connected component approaches the logarithmic behavior of a Moore graph \([3]\),

\[
L_{\text{rand}} \approx \frac{\ln N}{\ln \mu}, \tag{2}
\]

and the clustering coefficient approaches zero \([4]\),

\[
C_{\text{rand}} \approx \frac{\mu}{N}. \tag{3}
\]

On the other hand, regular networks are typically highly clustered, but at the price of having very large \( L \). Small-world networks lie in between these two extremes. They have large clustering, \( C \gg C_{\text{rand}} \), and small average shortest path, \( L \approx L_{\text{rand}} \), where \( C_{\text{rand}} \) and \( L_{\text{rand}} \) are the respective statistical quantities for a random network with the same parameters \( N \) and \( \mu \). From Table I, we see that the networks corresponding to all four programs we have studied are small-world networks. This result seems to be typically true for any large enough program. Therefore, we conclude that the logical structure of large programs can be described by small-world networks.

Notice that each source file corresponds to a totally connected subgraph in the network, since every header file included in a source file is connected to every other header file included in that same source file. Thus the network consists of several clusters (corresponding to the source files) interconnected by header files that are included in more than one source file. The clustering effect of the source files is the same as movies in the actors’ network (the “Kevin Bacon network”). Because of this, it is perhaps not surprising that \( C \) is large for our program networks. The fact that \( L \) is small, however, is not obvious and is due to nodes between otherwise distant clusters, caused in turn by header files included in more than one source file.

We have also investigated the influence of very highly connected nodes on the network, and how the networks’ statistical properties change if those highly connected nodes are removed. In order to do this, we define a new network from each of the four original programs by removing all the nodes with a number of links larger than \( N/4 \). The new networks will, of course, have smaller \( N \) and \( \mu \), and a larger \( L \). We now calculate \( C \) and \( L \) for these new networks. The results are displayed in Table II. We see that these networks still have the small-world property, in all cases. In fact, we have verified that the further removal of highly connected nodes always preserves the small-world property of the resulting networks, up to the point where we remove too many nodes, and the resulting networks are too small to define meaningful statistics. This shows that the small-world property in these networks is a robust phenomenon, and does not depend on the presence of a few highly connected nodes in the tail of the algebraic distribution \([1]\).

Finally, we observe that a network that contains full information about both header and source files can be defined. The result is a bipartite network \([29]\), which has two types of nodes (one corresponding to header files and the other to source files) and links that run only between nodes of different kinds, as defined by the “\#include” clause. The networks analyzed so far correspond to the projection of this bipartite network onto the space of header files. A similar projection with respect to the space of source files produces a network, whose nodes are source files and links are between source files that include a common header file. The network of header files and the network of source files share similar properties. In particular, both evolve according to a preferential growth and both exhibit the small-world feature.

In summary, we have shown that large computer programs correspond to growing networks that generally possess the small-world and scale-free properties. As computer softwares for various modern applications are becoming increasingly more complex, it is important to study and understand their topological structure for improved efficiency and improved performance. In particular, even for large computer programs the flow of information within the program is expected to be quite efficient because, as we have shown, in spite of the size of the program the average shortest path in the underlying network is very small. Also, some of the nodes of these networks appear to be much more connected than the average, which means that the corresponding files in the program are required for a large number of applications, making them relatively more important. This in turn, together with the very fact that different parts of the program (different applications) make use of a limited number of files, is expected to help the maintenance and debugging of the programs.
for example, the first files to be checked should be the most connected ones. We emphasize that our viewpoint that sophisticated computer softwares can be considered as networks is relevant because the network features identified in this paper are expected to be generic and universal.

This work was supported by the AFOSR CIP (Critical Information Protection) Program under Grant No. F49620-01-1-0317.

[1] A.-L. Barabási and R. Albert, Science 286, 509 (1999).
[2] A.-L. Barabási, R. Albert, and H. Jeong, Physica A 272, 173 (1999).
[3] B. Bollobás, Random Graphs (Academic Press, London, 1985).
[4] D. J. Watts and S. H. Strogatz, Nature (London) 393, 440 (1998).
[5] K. E. Stephan, C. C. Hilgetag, G. A. P. C. Burns, M. A. O’Neill, M. P. Young, and R. Kotter, Philos. Trans. R. Soc. London B 355, 111 (2000).
[6] M. J. Keeling, Proc. R. Soc. Lond. Ser. B 266, 859 (1999).
[7] A. L. Lloyd and R. M. May, Science 292, 1316 (2001).
[8] B. Kogut and G. Walker, Amer. Sociol. Rev. 66, 317 (2001).
[9] D. A. Fell and A. Wagner, Nat. Biotechnol. 18, 1121 (2000).
[10] M. E. J. Newman, Phys. Rev. E 64, 016131 (2001).
[11] M. E. J. Newman, Phys. Rev. E 64, 016132 (2001).
[12] M. Gitterman, J. Phys. A-Math. Gen. 33, 8373 (2000).
[13] U. Blum and L. Dudley, J. Evol. Econ. 11, 207 (2001).

[21] A. A. Gurtovenko and A. Blumen, J. Chem. Phys. 115, 4924 (2001).
[22] N. Guelzim, S. Bottani, P. Bourgine, and F. Kepes, Nature Genetics (London) 31, 60 (2002).
[23] A. E. Motter, A. P. S. de Moura, Y.-C. Lai, and P. Dasgupta, Phys. Rev. E 65, 065102 (2002).
[24] S. H. Strogatz, Nature (London) 410, 268 (2001).
[25] R. Albert and A.-L. Barabási, Rev. Mod. Phys. 74, 47 (2002).
[26] M. E. J. Newman, SIAM Review 45, 167 (2003).