Int. J. Man–Machine Studies (1991) **34,** 337–348

Liveware: a new approach to sharing data in social networks

IAN H. WITTEN
Computer Science, University of Calgary, Calgary T2N 1N4, Canada

HAROLD W. THIMBLEBY
Computing Science, Stirling University, Stirling FK9 4LA, Scotland, UK

GEORGE COULOURIS
Queen Mary and Westfield College, London E1 4NS, UK

SAUL GREENBERG
Computer Science, University of Calgary, Calgary T2N 1N4, Canada

(Received 1 May 1990 and accepted in revised form 1 August 1990)

While most schemes that support information sharing on computers rely on formal protocols, in practice much cooperative work takes place using informal means of communication, even chance encounters. This paper proposes a new method of enabling information sharing in loosely-coupled socially-organized systems, typically involving personal rather than institutional computers and lacking the network infrastructure that is generally taken for granted in distributed computing. It is based on the idea of arranging for information transmission to take place as an unobtrusive side-effect of interpersonal communication. Update conflicts are avoided by an information ownership scheme. Under mild assumptions, we show how the distributed database satisfies the property of *observational consistency*.

The new idea, called "Liveware", is not so much a specific piece of technology as a fresh perspective on information sharing that stimulates new ways of solving old problems. Being general, it transcends particular distribution technologies. A prototype database, implemented in HyperCard and taking the form of an electronic directory, utilizes the medium of floppy disk to spread information in a (benign!) virus-like manner.

1. Introduction

The information communication needs of individual computer users are frequently quite different from the large, highly-structured, shared databases that have been developed for corporate applications. Moreover, although many individuals now operate personal computers whose power rivals that of mainframes, they lack the infrastructure of support that accompanies larger computer installations.

Informal communication of non-critical information is vitally important for many information workers (Kraut, Egido & Galegher, 1988). The present paper asks how such information can best be communicated in an environment devoid of organized network support, at low cost, and with little effort for the user. We introduce an information distribution concept, called "Liveware", that is not so much a specific

0020-7373/91/030337 + 12$03.00/0 © 1991 Academic Press Limited

piece of technology as a fresh perspective on information sharing. Liveware stimulates new ways of solving old problems.

Here are three information sharing problems that are inadequately addressed by existing software and distribution mechanisms.

Problem 1: At the Apple kiosk of the human–computer interaction conference CHI'89, 1800 attendees had the opportunity to have their digital picture taken and enter information about themselves—interests, address, e-mail, etc. The database was to be distributed on CD-ROM one year later. Although interesting, the database is of limited value because no mechanism for correcting, updating or adding entries is possible due to the read-only nature of the medium. Much of the information will clearly become stale very quickly. But suppose the storage medium was writeable. The problem of tracking changes in individuals' information would be formidable. It is hard to see how any mechanism other than a sizeable central administration could permit such updates, and this would introduce such delays and overheads that people would probably not bother to contribute.

Problem 2: Consider the plight of those who have identical personal computers at both home and work. While trivial conceptually, the problem of keeping both filestores consistent is in practice horrendous. The common solution is to carry a floppy disk to and fro and transfer files manually. This poses a tremendous cognitive burden. If a disk or a file gets forgotten, the two filestores diverge. Problems escalate as the number of machines grows beyond two, or the number of people involved grows beyond one!

Problem 3: Recently a freeware Apple HyperCard document by Jakob Nielsen has been widely circulated. This interesting hypertext, described by Nielsen (1990), incorporates a number of attractive and unusual features. It records each person's track through the database in the form of a "history list", and users are requested to mail a copy of their disk to Nielsen when they have finished browsing, for analysis of usage patterns. But how many actually do? It would be better if the distribution mechanism retained old histories when the disk was passed on, to maximize the information gained on the rare occasions when a disk is actually returned. However, the history list is personal information, which raises the question of how to impose a degree of information security in a distributed environment.

These three scenarios illustrate different information distribution requirements. The first involves sharing personal information within a community, and necessitates distributed update. The second concerns the communication of information between machines controlled by one person. In the third, information is transmitted from the community to a particular individual.

This paper describes a technique that allows cooperation to take place in loosely-coupled socially-organized systems involving personal rather than institutional computers. The idea is technically very straightforward and involves a mixture of social convention and software support. It can solve diverse problems of information sharing, including those above, for which conventional techniques of data distribution would be extremely cumbersome to administer. Despite its simplicity, Liveware does not yet seem to have been put into practice. We describe a specific example of a Liveware system for an application very similar to Problem 1 above, though on a smaller scale. Sufficient implementation detail is included for the work to be replicated.

2. The idea of Liveware

The idea of Liveware is to arrange for information transmission to occur as a side-effect of interpersonal communication. It is designed for a communication environment, where connections are like chance meetings or casual encounters: they do not occur regularly and, when they do, maximum advantage must be taken of the opportunity to exchange information. This contrasts sharply with other protocols for computer communication, which are invariably predicated on the assumption that—except in cases of breakdown—information can be transmitted whenever the system finds a need to do so, an assumption which requires a permanent communications infrastructure.

2.1. DESIGN PRINCIPLES

Liveware was designed around three principles:

(1) *Symmetry:* exchange of information should always be two-way;
(2) *Transitivity:* users should act as carriers of other users' information; and
(3) *Transparency:* communication for information exchange purposes should be as unobtrusive as possible, and require negligible personal effort.

The first principle helps to ensure that maximum advantage is taken of every communication opportunity. When information is distributed in a conventional system, it is copied from source to receiver and no communication takes place in the reverse direction, except perhaps acknowledgements dictated by the protocol used. However, when connections are rare it makes sense to maximize the flow of information by transmitting information both ways. Both parties partially update each other's database, since in general each will have something to offer the other. This has the side-effect of simplifying the user's conceptual model of communication. It becomes symmetric: each party brings itself up-to-date whenever possible, and after an interchange both are in the same state.

The second principle is also intended to increase overall information flow. When communication opportunities are rare and fortuitous, it may be that the only way two parties can exchange information is through a chain of intermediaries. Interpersonal communication often takes advantage of this, although human fallibility makes it somewhat unreliable. Computer media are well suited to indirect communication because they can store information indefinitely and recall it accurately—information transfer is a transitive operation.

The third principle is intended to ensure that use is made of every available communication opportunity, in keeping with Kraut *et al.*'s (1988) requirements for successful casual interaction. It is most important that users are not tempted to turn off communication because it interferes with their own priorities. Although they may have nothing immediate to gain from the interchange, the quality of the information sharing service as a whole is maximized if every available opportunity is used. This indicates that use of the Liveware system should be as transparent as possible. The implications of these three principles are explored below.

2.2. MANAGING DISTRIBUTED INFORMATION

The information in a Liveware database is created and modified in a distributed fashion, at different sites and at different times. When two databases meet, a

symmetric update is performed to bring both to the same state. This merge operation, which forms the core of any Liveware system, should (whenever possible) be accomplished automatically to minimize intrusion.

To ensure that update conflicts can be resolved automatically, databases are split into units of information, each of which has a unique identification code, a single owner who alone can alter it, and a time stamp. Given this information, the minimal set of updates that are required to bring two databases to the same, updated state can easily be determined.

The way that Liveware works means that information ownership must apply regardless of whether the information is personal or not. Precisely one person must be responsible for each unit of information; that person is charged with ensuring that any update supersedes all previous versions. This is the only way—apart from manual intervention in the merging process—to circumvent the multiple-update problem where one update is overwritten by a different one made at a different place. Owners may make updates anywhere, to any version of the database, so long as they intend only the last change made to survive—for that will eventually replace all alterations made elsewhere. No locking of information is necessary (for owners can only be in one place at a time).

Some applications must disobey this information ownership rule and allow information to be updated by more than one person. For example, an address list may be shared by a group of users, and any address may be updated by any member. In this case the Liveware mechanism must flag duplicate entries during the merge operation and allow the user to decide how to treat them. Here the merging process supports a useful degree of data validation; if two or more users take responsibility for entering critical new information, any discrepancies are identified when their versions are merged.

In any case, additional access control rules might be desirable, depending on the situation. For example, in Problem 3 above where usage information is to be transmitted to the originator of a hypertext, read permission should be restricted to the information creator and the originator.

Liveware applications may permit new information owners to join in and contribute information anywhere, at any time. In this case it is necessary that owners are known publicly by their names to avoid subsequent ownership conflicts. It is insufficient merely to check that a new registrant's name is not duplicated in the database, for two people of the same name may register at different places and the problem is only detectable during merging, when the two versions meet. Hence full names, possibly supplemented to ensure uniqueness, should be used instead of abbreviated "computer style" log-in names.

2.3. THE COMMUNICATION ENVIRONMENT

We have so far deliberately avoided discussing the communication environment, because the idea of Liveware transcends any particular distribution technology. In our present implementations, the distribution device is a floppy disk. Users carry a disk containing their version of the database with them wherever they go. One user might visit another and plug in his disk, causing bidirectional information sharing; on returning to his own site he will insert the disk once again to update his personal computer. Ideally information transfer would not involve the user at all, but take

place as an immediate side effect of the disk being inserted into the drive. A group of users might meet and swap disks, or users might correspond by mail in an individual and uncoordinated fashion. Effective distribution relies on a rich interconnection structure where interchange occurs frequently, but it does not assume any particular organization. If people realize that their information is getting out of date, they will naturally make a special effort to communicate with others who are likely to be better connected. Social mechanisms will tend to ensure that information gets distributed as widely and as quickly as the need dictates, as they do conventionally in gossip circles and grapevines.

Of course, personal computer users exchange floppies at present. Unfortunately, manual procedures for updating files from floppy disks are notoriously unreliable. The difference with Liveware is that updates are performed entirely automatically, based on the timestamps associated with information units.

Liveware is in some respects like a virus, an observation that evokes both fascination and horror (Witten & Thimbleby, 1990). Indeed, the success and rapidity that has been observed in the spread of computer viruses—and the extreme difficulty of *avoiding* infection—provide a testament to the power of social networks to support information distribution. However, while Liveware systems strive to minimize any disruption caused by the fact that communication occurs, no attempt is made to conceal the fact that it occurs. Their use is discretionary: they only infect volunteers who want to share information.

Manual exchange of disks is by no means the only possible distribution mechanism. Others could be equally effective, or more so, depending on the means of communication available. Indeed, any way that can be used to spread viruses can be used to spread Liveware. For example, distribution could be piggybacked on e-mail or bulletin boards—the Liveware mechanism automatically determines minimal updates, so the bandwidth consumed need not be excessive. Or whenever a user connects to a distant host for remote login or file transfer, a mechanism could be invoked to see if advantage might be taken of that connection to update Liveware databases unobtrusively.

2.4. OBSERVATIONAL *VS* GLOBAL CONSISTENCY

A shared database must be sufficiently timely to remain relevant for shared work. If users do not communicate their information with others, the database will inevitably get out of date and different versions of it will disagree. If you really need guaranteed consistency, you must pay for the infrastructure it requires. For many purposes people will be prepared to trade currency off against cost—particularly the ongoing administrative and communications cost associated with conventional distributed databases. Liveware is still considerably better than a paper file or card index; the fact that paper has severe problems—and none of the advantages of automatic updating—has never stopped people from using it effectively. Liveware is certainly good enough for many applications: it does a far better job than conventional media, and at a far lower cost than conventional computer solutions.

Liveware enjoys the property of *observational consistency*. So long as users carry their information around with them, they cannot observe inconsistency in the database. This property holds when no one travels faster than their information, as

certainly happens if they are using e-mail or carrying it in their pocket on a floppy disk. As soon as a user encounters an inconsistent database, Liveware sorts it out before he can use it. Although one may *believe* the database to be inconsistent, one can never *observe* it.

Observational consistency can also constitute a limited security measure. Consider automated teller machines (ATMs) that place cash withdrawal limits on their customers. During periods of heavy load (e.g. over lunch time) these machines run in batch mode. On the assumption that a customer can only be in one place at a time and that the time to get from one machine to another exceeds the period that the machine is in batch mode, customers cannot overdraw their limit. They cannot tell if the system is batch (like Liveware) or real time (fully distributed). Of course, criminals copy cash-cards, and organized groups can masquerade as a single customer in many different places and withdraw many times the cash limit.

Whether Liveware is any good for a particular purpose depends on whether or not observational consistency is adequate for that purpose. In many information sharing situations, real global, moment-to-moment consistency is either unnecessary, or a luxury that will not be missed. Besides, on those occasions when consistency is required, a group of Liveware users could easily arrange for a disk to be passed around the group twice—the first pass collecting the information, the second distributing it. This procedure guarantees total consistency so long as updates do not occur while it is taking place (though often it will be overkill).

In general, global as opposed to observational consistency is only required when a group of people are working on a single object that is visible to the world at large, for example, a monolithic report. Observational consistency is always sufficient when users are working collaboratively but "doing their own thing"—as when members of a group are assigned their own tasks and it is not important whether other members observe them performing those tasks straight away.

3. Example: an electronic directory

Consider Problem 1 from the Introduction, namely to establish a database that records information about each member of an interest group. In fact our example concerns the community of human-computer interface researchers in Scotland; the purpose of the database is to facilitate cooperation by ensuring that at least people will know what others are doing, and how to get in touch with them. Setting up a conventional database would require identifying the members of the group, soliciting information from them all, encouraging them to respond, collecting data, collating and distributing it.

Some of the clerical work could be avoided by circulating a questionnaire on disk and encouraging interested people to pass it on to colleagues, returning completed forms for central, automated collation. Furthermore, if collation can really be automated, it can take place at any site, not just centrally—this is the idea of Liveware.

A Liveware database can act as both questionnaire and information directory. People fill it out and pass it to others who may be interested. As it spreads it accumulates useful information from everyone who contributes. When run on a machine that already has a copy, the databases merge and update each other as

appropriate. The need to return disks is eliminated, for collation is automatic. It is not even necessary to identify the target group in advance, for people can pass the system to their colleagues and anybody can join in. Updates can be done anywhere, at any time, and will percolate throughout the community. This scheme has been implemented as a HyperCard stack.†

3.1. THE DATABASE

Viewed as an ordinary hypertext document, the Scottish HCI Database is a standard HyperCard application with two principal components: personal records of participants; and an electronic noticeboard.

Each person has a card (like that shown in Figure 1) on which he or she records name, address, phone number, email address, and a list of one-line phrases describing research interests. An **Add a new person** button on the front page (not shown) allows new owners to register with the database. They enter a dialogue that solicits their name and initial password, and a blank card is created that they can fill out.

The second component is an electronic noticeboard on which any user can post notices. Entries have an expiry date after which they are automatically deleted. A facility is provided so that individual notices may be hidden: having been seen by one user they will not be shown to him again, but are retained in the database and will be passed on to others.

Two indexes are maintained automatically. A list of all people represented is collected on a separate card; clicking on an entry brings up that person's card. A

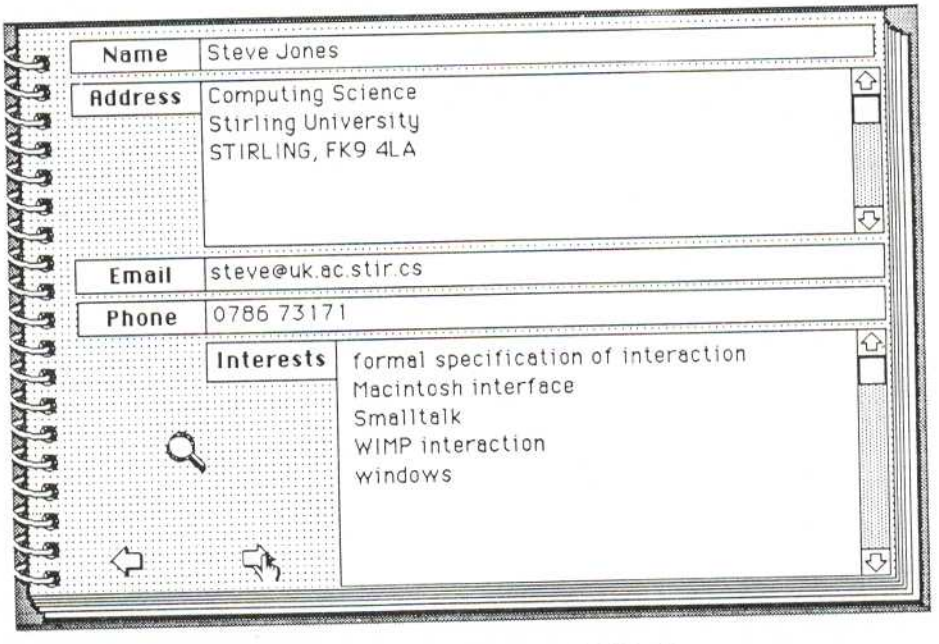

FIGURE 1. An entry in the Scottish HCI Database.

† HyperCard is a product of Apple Corporation (Apple, 1987). It has the advantage of being oriented towards the end user, and is widely available because it comes free with all Macintosh computers.

complete list of every person's interests is automatically compiled on another card; clicking on a line of this index cycles through the cards of people who have declared that interest. The stack contains a few cards owned by the stack creator that give information and help about the database and about Liveware, and a brief summary is included of the social conventions on which the system relies.

What has been sketched so far is a standard application of hypertext. The following sections describe the Liveware component, beginning with the means of enforcing information ownership—which, as noted above, is absolutely necessary for distributed update to work with fully-automatic merging.

3.2. SECURITY

A new pulldown menu, shown in Figure 2, is added to the HyperCard menu bar to give controlled access to the hidden Liveware control information. It allows the owner of particular cards within the database to log in (called "unlock" in the menu), change password, or change the expiry date of those cards he owns.

Passwords are used to impose a degree of integrity on the information. They are encrypted on entry by HyperTalk's† built-in "ask password" facility and stored (invisibly) on each card. The same password is stored on all cards that belong to a given owner; thus cards are self-contained and can be treated independently. In practice, the security mechanism can be circumvented fairly easily by anyone acquainted with the HyperTalk language, since the code is stored in source form and can be read by all. Although more complex schemes could be implemented, real protection is simply not possible in a distributed system without hardware support.

3.3. THE LIVEWARE COMPONENT

The merge operation that constitutes the kernel of Liveware is implemented entirely in HyperTalk. Whenever a new disk is inserted, or another Liveware database is found, a merge should occur. Because the system is experimental, merging does not occur autonomously but takes place under user control.

The Liveware control card, illustrated in Figure 3, contains three dialog boxes. The field entitled "Versions found" displays the names of other HyperCard stacks that are versions of this database. Although it is quite feasible to scan a floppy disk automatically to check for versions of the database, scanning a hard disk can take

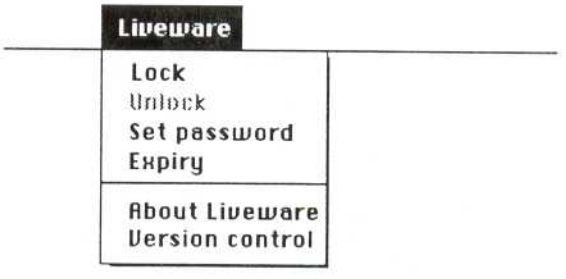

FIGURE 2. The Liveware menu.

† HyperTalk is the language of Hypercard.

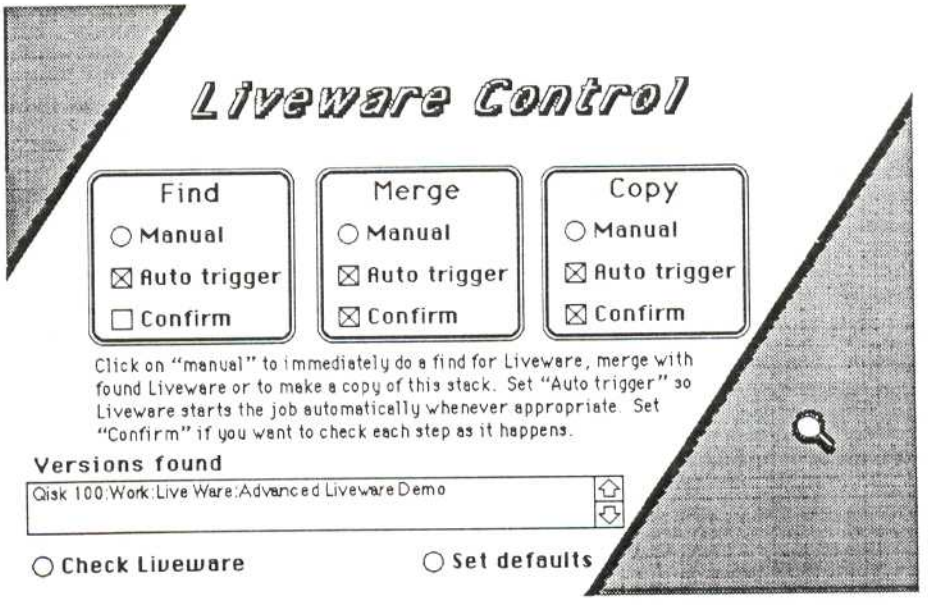

FIGURE 3. The Liveware control card.

some time. To avoid inordinate delays whenever the system is entered, users can add new names to the list manually with the "Find" dialog box.

The "Merge" dialog box permits one to initiate the merge operation manually. Alternatively it can be triggered automatically whenever a new version of the database is entered into the "Versions found" field. The "Copy" box allows the user to make new copies of the database without leaving HyperTalk—we want to encourage copying as much as possible. Checking the "Confirm" box will request user confirmation before carrying out auto-triggered actions.

3.4. THE HYPERCARD IMPLEMENTATION

The information necessary for Liveware to operate correctly is called the *Livestamp*. Normally of no especial interest to the user, it is stored in a hidden field. Different forms of Liveware have different methods for merging, and hence different Livestamp requirements. Each card in the Scottish HCI Database requires the following information within its Livestamp.

- *Signature:* A code unique to the owner of the card. It consists of the owner's actual name, combined with a machine-generated identifier (to "uniquify" the name so that collisions can at least be detected during the merge operation) and an encrypted password (to avoid impersonation).
- *Identification code:* Each card (for a given signature) has a unique code—easily provided by the computer's clock.
- *Expiry date:* Used to destroy the card when it has exceeded its useful life.
- *Dormancy flag:* Set to true if the current user is not interested in the card, but nevertheless wants to share the information with other users. (The alternative is to delete the card: then, of course, it cannot be shared.)

- *New flag:* Set to true if this card has not yet been seen. It enables the user to locate newly acquired information easily.
- *A time stamp:* The most recent time when the card was created or last modified.

In addition to this information maintained in a card's Livestamp, there is other information required only once in the database stack:

- *Deletion list:* A list of cards, previously shared, but which have since been deleted, either because of explicit owner action or because of expiry dates being brought back.

Finally, there may be additional information in the Livestamp to support mechanisms specific to the application. For example, "fixed" cards like those giving information about the purpose of the stack are treated separately, and are owned by the stack's creator. Although the Scottish HCI Database permits anyone to register as a new user, the stack creator alone (or another nominee) could be empowered to introduce new users and to reset passwords for owners who have forgotten them, or who have been withdrawn from the stack.

3.5. THE SCOTTISH HCI DATABASE IN USE

A Liveware form of the Electronic Directory was introduced to the Scottish HCI community in the spring of 1990. Users reported that Liveware is successful in what it tries to do, and it reached over 70% of the target audience within the first few weeks. Its replication over this short time was clearly effective, matching the exponential behaviour expected of a computer virus.

Longer term use of the Electronic Directory was not as promising. Users now report that since the directory information in the database has become relatively stable, there is insufficient motivation to carry it around at all times (except for larger organized meetings). This in turn reduces penetration of the database into the remaining unreached community.

This preliminary experience suggests that Liveware as a technology can work (as seen by its initial rapid growth), but that its overall success or failure will depend upon the cost/benefit trade-offs perceived by users. In this case, the Electronic Directory failed to live up to the *principle of transparency*, which demands that information exchange be unobtrusive and require negligible effort. While the early rapid growth of the database gave community members enough incentive to update their database frequently (and thus update other databases), the diminishing returns given to them in the older and relatively stable system made the effort of carrying the database around too costly.

4. Controlling resource usage

Replicated databases can be expensive to store, and Liveware databases tend to grow monotonically in terms of the data they contain (although stability may set in once the target population of users has been reached). There is no doubt that Liveware is intrinsically resource-hungry. However, its growth can be controlled—at least to some extent—in three ways: deleting information; filtering information; and restricting the user community.

It is easier to add information to Liveware than to remove it. Deleting a unit of information locally has only a temporary effect, for unless it is removed in concert

from all versions of the database it will eventually be restored through the merging operation. Instead a record must be kept of the unit's identification so that it can be deleted whenever it appears again. A different mechanism for permanent deletion, also illustrated in the Scottish HCI Database, is to furnish information units with expiry dates. Another, inspired by Jefferson's (1985) notion of "virtual time", is to arrange for cards to be chased by "anti-cards" that annihilate them. This could perhaps be expedited by having Liveware record the immediate recipients of information in order that anti-cards might seek the same route, although we have not seriously considered such protocols.

So far, we have assumed that each copy of the database ideally contains the same information. Instead, a scheme of user profiles could be implemented to permit greater selectivity when picking up and dispensing information. The merge operation would require the user to log in and would then respect his profile, inserting only those cards that match it. Extensive use of profiles will mean that the spread of information is socially moderated by the group as a whole: what people are interested in will disseminate rapidly, specialist information will not propagate. The requirement of automatic merging means that users will have to *describe* the information they want, rather than look for it directly; indeed the question of specifying user profiles automatically is a topic of current research (e.g. Malone, Grant, Turbak, Brobst, Cohen, 1987; Chen, Ekberg & Thompson, 1989).

Systems like the Scottish HCI Database that let new users register and contribute in a completely uncontrolled manner are likely to become polluted by unwanted users. Fortunately, Liveware can be used to implement elaborate schemes for club membership. For example, existing card-owners may be empowered to introduce new ones, or several may have to collaborate to propose a new one. Liveware may enforce collaboration in a single interactive session, or permit nominations to be stored on cards owned by the proposers to distribute the process in time and place. In the latter case the nominee could take his Liveware disk round potential proposers until he has collected the requisite number of nominations. Moreover, each owner's nominations may be stored to allow limits to be placed on the number of introductions that owners may participate in. All of these possibilities are quite simple to implement; the chief problem is in deciding which scheme is suitable for any particular purpose.

Whenever physical interchange of disk is the transmission medium, Liveware will be limited by the storage capacity of common interchangeable media. While it may never be suitable for really large databases, it seems likely that current advances in removable optical disk storage (Freese, 1988) will open up a wide range of potential applications.

5. Conclusions

A method has been described that allows information sharing to take place in loosely-coupled socially-organized systems involving personal computers. Conventional means of information sharing on computers are expensive. Liveware, in contrast, is cheap and intrinsically intertwined with social conventions of spreading information. A practical and easy-to-use prototype has been implemented in HyperCard and was recently introduced to the Scottish HCI community; it is still

too early to document its course of evolution. More generally, the idea has many applications, from public domain databases, through facilitating the exchange of information in interest groups, to providing identical environments on several computers controlled by a single person.

We believe that the idea of casual information sharing with low administrative overhead is timely and reflects many human to human interchanges. The increasing use of laptop portable computers underscores the need for effective, yet informal, mechanisms of multiway information distribution, as does the burgeoning complexity of computer systems. For example, Liveware might permit users to automatically pick up updates to software. How many times have you wished you had the most recent bug fixes from the supplier, or even bug reports from other users? Liveware is a step in the right direction.

The solution we propose has the technical merit of replicating the database on as many machines as are involved (and on all disks used for transport); it has the economic merit of costing nothing and requiring no wiring or other installation. Because there need not be any technical infrastructure such as networking, nor the usual geographical restrictions of networks (e.g. being in a single building), the sharing mechanism meets the needs of many mobile and flexible social work groups.

It is a pleasure to acknowledge valuable input by Stuart Anderson, Ann Burnie, Jean Dollimore and Steve Jones.

References

APPLE COMPUTER, INC. (1987). *HyperCard User's Guide*. Cupertino, CA.

CHEN, J. C., EKBERG, T. W. & THOMPSON, C. W. (1989). Querying an object-oriented hypermedia system. In *Proceedings Hypertext II*, University of York, UK.

FREESE, R. P. (1988). Optical disks become erasable. *IEEE Spectrum* **25,** 41–45.

JEFFERSON, D. R. (1985). Virtual time. *ACM Transactions on Programming Languages and Systems* **7,** 404–425.

KRAUT, R., EGIDO, C. & GALEGHER, J. (1988). Patterns of contact and communication in scientific research collaboration. In *Proceedings ACM Conference on Computer-Supported Cooperative Work*, Portland, Oregon, September, pp. 1–12.

MALONE, T. W., GRANT, K. R., TURBAK, F. A., BROBST, S. A. & COHEN, M. D. (1987). Intelligent information-sharing systems. *Communications of the ACM* **30,** 390–402.

NEILSEN, J. (1990). The art of navigating through hypertext. *Communications of the ACM* **33,** 296–310.

WITTEN, I. H. & THIMBLEBY, H. W. (1990). The worm that turned. *Personal Computer World*, July, 202–206.