On the impact of network latency on distributed systems design

Jesper M. Johansson
MIS Department, Boston University, 595 Commonweath Ave., Boston, MA 02215, USA
E-mail: jjohanss@bu.edu

Research in distributed database systems to date has assumed a “variable cost” model of network response time. However, network response time has two components: transmission time (variable with message size) and latency (fixed). This research improves on existing models by incorporating a “fixed plus variable cost” model of the network response time. In this research, we:

(1) develop a distributed database design approach that incorporates a “fixed plus variable cost”, network response time function;

(2) run a set of experiments to create designs using this model, and

(3) evaluate the impact the new model had on the design in various types of networks.

Keywords: distributed database systems, network design, network response time, latency, parallel processing

1. Introduction

In a study of 108 members of the Society for Information Management (composed largely of senior information systems executives) building a responsive (information technology) infrastructure was rated the most important information systems management issue facing corporations [1]. In addition, developing and managing distributed systems was rated number three, and planning and managing communication networks was rated fifth. In another 1996 study, management of corporate communication network services is shown as the number one core firm wide IT infrastructure service [2]. Total spending on all telecommunications in the United States was almost $300 billion in 1996, and spending will likely increase by over 10% per annum for several years ahead. Much of that spending is for distributed information systems.

The basic component of these distributed information systems is a Distributed Database Management System (DDBMS). In spite of the fact that DDBMSs are now becoming commonplace, our understanding of their design and the interactions between their various components is based on outdated technology. This research aims to investigate the impact of the network infrastructure on the design of distributed information systems.

A Distributed Database Management System is a fundamental portion of a distributed information system. It is a set of data management applications designed specifically for managing organizational data that is dispersed among the nodes of a computer network. Design of distributed databases (DDBs) differs from design of centralized databases (DBs) in three fundamental aspects:

(1) Incorporation of network design issues in the database design process.

(2) Allocation of data to sites across a network, including data fragmentation, replication and synchronization.

(3) Allocation of data retrieval and processing activities to the sites.

These aspects of DDB design are highly interrelated. For example, the preferred fragmentation depends on the optimal fragment allocation and on the processing schedules of the queries that access the data. However, processing schedules depend on the location of data. Furthermore, network parameters influence both data allocation and processing schedules. The problem of data allocation by itself has been shown to be NP-Complete [3]. Process allocation proceeds similarly, and is also NP-Complete.

2. Previous research

The early distributed database research focused largely on the allocation of entire files in a network [5], and on the complexity of the problem [3]. Several authors pointed out the similarities between the distributed database problem and more general operations research problems (cf. [3,5]). Consequently, much of the research in the area has made use of linear optimization techniques. However, Casey pointed out that although the linear approaches may give an adequate first-order approximation of the solution, "the cost of shipping information may not be linear with the amount sent" [6]. Recent research efforts on DDB design frequently use an iterative optimization approach alternating between fragment allocation and query processing schedules [7–10]. The importance of taking into account both the data and the query processing schedules is pointed out in, among others [9,11]. In the work we propose, we will build on the model developed in [9,10]. This model uses an iterative optimization of data allocation and query optimization based on a nested genetic algorithm. The genetic algorithm serves to define the sample of possible solutions which are tested.

Much of the previous research on DDB design focus on DDBs running on Local Area Networks (LAN) (cf. [12–14]). Unfortunately, it is rarely worthwhile to distribute a database on a LAN. The geographic coverage of a LAN is not sufficient to create variations in network response time. Parallelism, often cited as a primary reason behind DDBs, is better achieved using a massively parallel computer or symmetric multiprocessing. In addition, LAN protocols commonly use broadcast techniques – all messages are transmitted to all nodes, regardless of data distribution. In broadcast networks, therefore, minimization of network response time by data distribution is meaningless.

3. Network response time

The technology used in corporate computer networks has become much more powerful in the last 10 years. Until the 1990's, the dominant type of Wide Area Network (WAN) infrastructure was low-speed point-to-point leased lines. These lines had very low data rate, typically ranging from 9600 bits per second (bps) to 56 Kbps (thousand bps), with limited use of T-1 service at 1.544 Mbps (million bps). Today, typical WAN services range from 64 Kbps to several Mbps. Even higher speeds will be common in the near future. At these speeds, the dominant factor influencing network response time (NetRT) is not the data rate, but the latency. Latency is fixed with respect to data size and network data rate, but varies with distance. Higher data-rate networks incur the same latency as lower data-rate ones over the same distance. Therefore, the faster the network, the more significant latency becomes. To quote ([4, p. 346]): "At (high) speeds we are latency limited, not bandwidth limited".

3.1. Transmit time

This research deals with transmissions at the transport layer. Typically, the computer can process data faster than the network can accept it. Therefore, the bottleneck below the transport layer is the network capacity (or data rate), which results in transmit time – the time it takes to transmit data onto the medium. E.g., if network capacity is 56 kilo bits per second (kbps), and a process needs to transmit 140 bytes, it would take $140 \times 8/56000 = .02$ seconds to transmit the data. Transmit time only accounts for the time it takes to send the data; "putting it on the wire" so to speak. Therefore, it is a function of the amount of data transmitted and the network capacity (bandwidth).

Transmit time is the network factor that has been best covered in DDB design literature. Most research to date assumes that transmission time between any two sites is identical. In other words, the network is homogeneous (cf. [7,11,15–18]). A more realistic situation is where the links between nodes are allowed to differ in capacity. The models developed in [6,9,10] allow this.

Many authors attempt to minimize total dollar cost of the DDB. In such an analysis, the cost of transmission, rather than transmit time, is measured [12,19,20] and [11,21]. All work on cost minimization has assumed cost models which are linear with respect to data size. All models also consider variable cost only. The model presented in [8] perform cost minimization, and allows the cost of one unit of data transmission to vary between links.

In a practical situation, it is extremely difficult to assign dollar costs to the various components in a DDB design. When companies purchase network services (and computer services and equipment as well for that matter) they buy a fixed amount of capacity. If additional capacity is needed, the next highest available increment of capacity is purchased. For example, in June 1998, companies could purchase a 1.544 Mbps Frame Relay drop for just under $2000 per month, with a $3000 fixed startup cost. Should more capacity be needed, sustained usage service can be purchased in increments of 1.5 Mbps, for between $2000 and $5000 per month per 1.5 Mbps increment. Those prices do not include necessary access circuitry. These characteristics also apply to other components of a DDB, such as storage and servers. The cost functions involved are not linear with the amount of data transmitted, but are step functions. This makes cost minimization less realistic and much more difficult. Therefore, response time minimization is a more tractable objective when designing DDBs. Cornell and Yu developed a model which performed response time minimization [17]. Rho developed models for both cost and response time minimization [9].

In the context of query optimization, Hevner and Yao [22] is worth a special mention. That article presents an approach which uses a network response time model that incorporates a variable cost, based on the volume of communication, and a communication startup cost. This communication startup cost represents connection establishment time. However, connection establishment time is dependent on the communication protocol used. Communication protocols for transaction systems (such as a DDB) are typically more efficient if they are connectionless, or if they can transmit the transaction information in the connection setup message [4]. For example, a transaction oriented version of TCP/IP is currently under development [23]. In such protocols connection establishment cost is zero.

3.2. *Queuing delay*

Transmit time may also have a contention factor if several concurrent transmissions are competing for access to the same network. This contention delay is dependent on the volume of communication from all computers connected to a certain network link. It can be thought of as a queuing delay before a given process is granted permission to transmit. (Generally, in WANs, it manifests itself as a decrease in the available bandwidth, but it is analytically convenient to deal with it as a queuing delay.) As the network gets faster, queuing delay becomes smaller. For very fast networks, the queuing delay becomes negligible.

Most research to date has ignored contention between transmissions. Exceptions include [9,10,17,18], all of which treat contention as a queuing delay as described above. In addition, the LAN design presented in [13] takes into account contention. However, the purpose of that work is to design a local area network architecture, not a database, so its relevance to distributed database design is limited.

3.3. *Latency*

Latency is the time it takes a signal to propagate from sending node to receiving node once it has been transmitted. It is calculated as the time differential between when the first bit is transmitted and when it is received. Latency, in its simplest form, depends on the distance between sender and receiver and on the communications protocol and techniques used. If no intermediate devices are used which add delays, latency becomes the inverse of signal speed (SS). The theoretical upper limit on SS is the speed of light. However, in wired networks SS will always be lower. For example, the propagation speed of an electrical signal in a copper wire is about 2×10^8 meters per second, or about 2/3 the speed of light ([24, p. 119]). Protocol and technology considerations may further increase latency, as will intermediate switching and repeating devices present on the network link. For the preliminary results presented later, we use 2/3 the speed of light as the signal speed, since it represents a best-case scenario of latency in networks based on copper wire.

Researchers in DDB have largely ignored latency, although [25] considers latency when performing file migration on a LAN, and the network designed in [13] does incorporate latency considerations. A primary motivation of the research presented here is to incorporate latency in the network response time function used for DDB design.

4. Data transmission process

Figure 1 illustrates the process of transmitting a message over a transmission link. Between time index 0 (t0) and time index 1 (t1) the process is idle. It is being queued, and is experiencing a queuing delay. At time index 1 the sender begins transmission. The sender is transmitting onto the link until the entire message has been transmitted. This occurs at t2. The time difference between t1 and t2 represents the transmit time. The receiver does not begin receiving data until t3. Since we measure latency from the time the first bit is transmitted to the time the first bit is received, latency is the difference between t1 and t3. This may seem counter-intuitive since it seems to overlap with the transmit time. However, we must keep in mind that at time t3 the receiver has only received the first bit. The time from when the receiver receives the

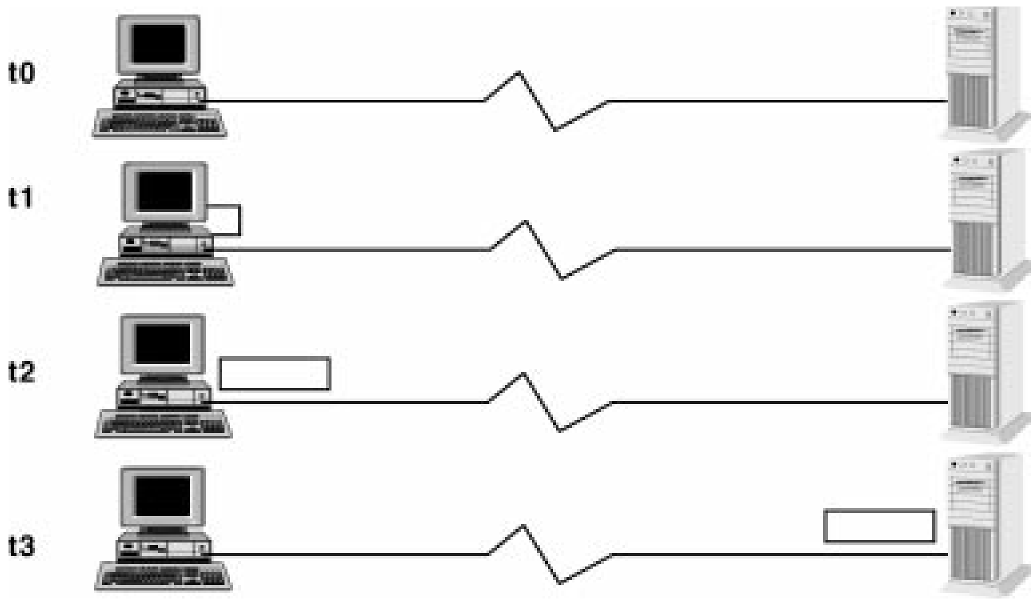

Figure 1. Illustration of data transmission.

first bit to when the last bit is received is equivalent to the time it took to transmit the message. Thus, the total time passed from the time the sender is ready to receive until the receiver has the entire message and can begin processing is the queuing delay (t1 – t0) plus the transmit time (t2 – t1) plus the latency (t3 – t1)

To highlight the importance of latency in this sum, consider a 140-byte message, transmitted for 600 miles. If we assume a signal speed of $2/3$ the speed of light, the transmit time in a 56 Kbps network is 20 milliseconds (msec), while latency is 5 msec. By contrast, in a T1 network, transmit time is only 0.7 msec, while latency remains 5 msec. In the 56 Kbps network, we would have underestimated NetRT by 20% if we disregarded latency. In the T1 example, NetRT would have been underestimated by nearly 88%! It is important to realize that the effect of not considering latency is related to message size. Small messages are more sensitive than large ones. This stems from the fact that latency is orthogonal to message size, and thus accounts for a proportionally larger component of NetRT for small messages.

5. Network response time model

This research extends the work in [9,10]. Those authors used a pure "variable cost" response time model, which, as discussed above, may be sufficient for low-speed networks, but not for high-speed networks.

In this research, we develop a new evaluation algorithm based on a "fixed plus variable cost" response time model. The value of the approach is demonstrated by solving a series of problems and comparing the solutions generated using a "variable cost" response time model with those generated using the "fixed plus variable cost" response time model. The new model of response time incorporates latency as a determinant of NetRT. The average response time equation due to network communication proposed by [9,10] is shown in equation (1):

$$R_{COM}(k) = \sum_t \sum_p \sum_m \left[\frac{\left(\frac{TL(t,p)}{W(t,p)}\right) N(k,m,t,p)}{\left(\frac{UL(t,p)}{W(t,p)}\right)^2 - \left(\frac{UL(t,p)}{W(t,p)}\right)\left(\frac{TL(t,p)}{W(t,p)}\right)} + \frac{H(k,m,t,p)}{UL(t,p)} \right]. \quad (1)$$

The first term in this equation determines the queuing delay, based on the assumption that the requests for data transfer on the network follow a Poisson distribution. The service times on the network may follow a general distribution. The term $N(k,m,t,p)$, is a binary value which is 1 if step m of query k requires communication on the link between t and p, 0 otherwise. The second term tracks transmission time. By summing over all links, it will track total transmission time. In light of the fact that latency is a considerable component of NetRT, it is clear that this equation does not adequately capture the dynamics of NetRT. Therefore, a new equation for NetRT is needed.

$$R_{COM}(k) = \sum_t \sum_p \sum_m \left[\frac{\left(\frac{TL(t,p)}{W(t,p)}\right) N(k,m,t,p)}{\left(\frac{UL(t,p)}{W(t,p)}\right)^2 - \left(\frac{UL(t,p)}{W(t,p)}\right)\left(\frac{TL(t,p)}{W(t,p)}\right)} + \frac{H(k,m,t,p)}{UL(t,p)} + [L[D(t,p)]]N(k,m,t,p) \right]. \qquad (2)$$

Equation (2) shows the new model of average NetRT. Two new terms are added: L representing the latency per unit distance, and $D(t,p)$, representing the distance between nodes t and p. The term $N(k,m,t,p)$ is defined as before. This equation is based on the assumption that latency is directly related to distance. This is a reasonable assumption if we consider the components of the network which add to latency. The first is the wiring, and the amount of latency added is proportional to the distance. Second, we have repeaters (or amplifiers in an analog network). These are evenly spaced with a certain distance between them. Therefore, their delay too is proportional to distance. Thirdly, we have switches. However, in this work, we are assuming that the network is directly connected, or that all connections are switched through one of the other nodes. Doing so would create a transmission on a different link which is evaluated in the context of that link instead. In modern networks, there may be other kinds of switches which can switch partial messages and thus add much less of a delay. In the current research, we are concerned primarily with evaluating whether latency has an impact. Therefore, those kinds of switching issues are disregarded at this point.

6. Experimental design and initial results

We have performed an initial study comparing the new model of NetRT to the old version. We designed a program to evaluate a homogeneous network with a very small problem and a distance between nodes roughly equivalent to that between New York and Los Angeles. The database in question has 5 data fragments (numbered 0 to 4), 6 retrieve queries and 11 update queries (numbered from 0 to 16) and 3 nodes (0 to 2). We assume that the network is fully connected, and that the database servers are connected directly to the backbone. Therefore, switching costs are disregarded.

Queries and updates occurred with varying frequency, some very often, others very seldom. The fragment by node matrix in table 1 shows the queries run on various fragments, and the location from which they were run. R stands for restrict, J stands for join, and U for update. Q_n means query n.

Table 1 provides an overview of where operations on the various fragments occur. All else being equal, we should expect that fragments might be allocated to the nodes where retrieve operations occur. Thus, fragment 0 could be expected to be allocated to node 1; fragment 1 would be expected to be replicated to both node 0 and 2, and so

Table 1
Fragment by node query matrix.

Fragment	Node		
	0	1	2
0	(U: Q6)	(R, J, J: Q0), (R: Q5), (U: Q14)	
1	(R: Q2), (U: Q7)		(R, J: Q4), (U: Q13)
2	(U: Q8)	(R, J, Q0), (U; Q16)	(R: Q3), (U: Q12)
3		(U: Q9)	(R, J: Q4), (U: Q11)
4	(R: Q1), (U: Q15)	(R, J: Q0), (U: Q10)	

on. We can use this as a rudimentary validation of the allocation algorithm. However, other factors may impact the allocation, as we shall see later.

This problem was evaluated using exhaustive enumeration in order to ensure that the optimal solution was found in all cases. We evaluated all possible data allocations, and then for each data allocation, we evaluated all possible operation allocations to find the optimal solution. It is necessary in an exhaustive enumeration situation to restrict the problem size severely. We deemed this problem the largest feasible, while still adequate for evaluating the theory. These initial experiments followed the approach taken in previous DDB research and did not consider parallelism between transmissions.

The network was evaluated at three transmit speeds:

(1) 56 Kbps, representative of WANs of the 1980s and early 1990s.

(2) 1.544 Mbps, T-1 class network, representative of WANs of the mid-to-late 1990s.

(3) 44.736 Mbps, T-3 class network, representative of extremely high-speed networks emerging as WAN choices in the late 1990s.

7. Results

The first result to look at is whether the time spent on each of the NetRT components in the optimal solution differed depending on whether or not latency was considered. This will tell us whether the optimal solutions differ, and thus whether latency is an important factor to consider. If, for example, the transmit time was different between the latency and the no latency case for the same network, this would be an indication that latency indeed had an effect. Table 2 presents the total time spent on each of the network response time components over the length of the evaluation period, summed over all queries and network segments. Thus, the figure of 1065 for queuing delay in the first column indicates that a total of 1065 seconds was spent on queuing delay by all queries over the 15-day evaluation period, regardless of whether latency was considered.

As we can see from table 2, the introduction of latency into the model had no effect in the 56k network. Latency was a relatively minor factor. However, it is

Table 2
Total time spent on the NetRT components.

	56K		T1		T3	
	No latency	Latency	No latency	Latency	No latency	Latency
Queuing delay	1,065	1,065	1.38	3.63	0.02	0.02
Transmit time	81,769	81,769	2,966	3,887	247	234
Latency		7,435		6,300		4,728

interesting to note that it is more important than the queuing delay. However, neither the queuing delay nor the transmit time changed as a result of considering the latency. This means that the existing models may be accurate enough for slower networks.

In the T1 case, we immediately notice two things. The first is that both the queuing delay and the transmit time changed as a result of the introduction of latency. This indicates that latency had an effect on the allocation, which we shall discuss later in the paper. The second thing to notice is that not only is latency now the major factor, the queuing delay is also extremely small. This may indicate that we could make the model more parsimonious and less complicated by omitting the consideration of queuing delay.

In the T3 case the effect of latency is even more pronounced than in the T1 case. It now accounts for the vast majority of the total NetRT. This means that neglecting to consider latency would most likely lead to profoundly erroneous results in a very fast network. It is clear that the allocation of data must have changed due to the proportionally larger latency component in the faster networks. Figure 2 illustrates the proportion of total network response time due to the fixed latency component. Note that figure 2 is a semi-log plot.

Table 3 shows the optimal data allocation. Under each case within a network is a 5 $\times$ 3 matrix representing the allocation. Rows represent fragments while columns

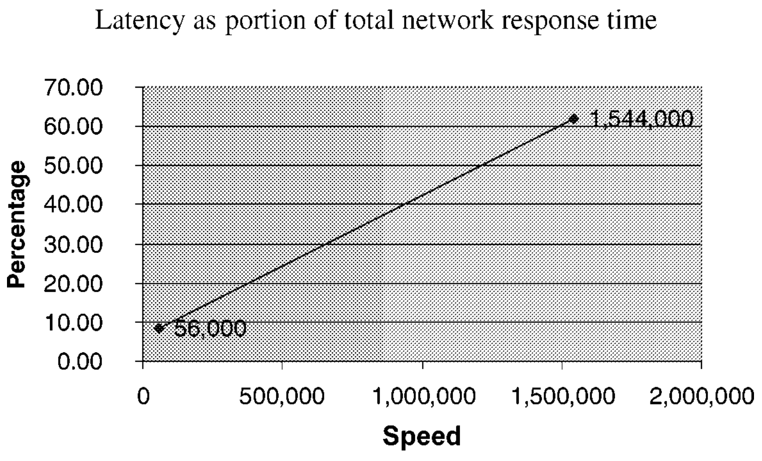

Figure 2.

Table 3
Optimal data allocation.

	56K		T1		T3	
	No latency	Latency	No latency	Latency	No latency	Latency
Fragment allocation	010	010	010	010	110	010
	001	001	001	001	001	001
	011	011	011	010	011	010
	001	001	001	001	001	001
	110	110	110	110	100	010

represent nodes. Thus, a row-column intersection of 1 means that the fragment represented by that row is allocated to that node in the optimal solution. E.g., the value of 010 in the first row under 56K No Latency means that in the optimal solution, found without considering latency under a 56K network, fragment 0 was allocated only to node 1.

As we suspected, the allocation did not change as a result of the consideration of latency in the 56k case. This provides further evidence that existing models performed adequately when used to allocate data on slower networks. In the T1 case, however, the allocation did change. Predictably, the increased proportional cost of small messages caused the degree of replication to be decreased since fragment 2 was no longer allocated to node 2. Thus the cost of the many small messages involved in updating a fragment in two locations outweigh the cost of retrieving data from a remote node when it is needed at node 2. This results in a decrease of the level of replication in the system.

In the T3 case the replication was decreased even further, to the point of not replicating data at all. The NetRT becomes virtually a fixed cost in this situation and as a result, we do not replicate any data at all. However, there is another interesting lesson to be learned from table 3. If we refer back to table 1, we see that fragment 0 is never accessed from node 0. Yet, fragment 0 is allocated to that node in the optimal solution if we do not consider latency on a T3 network. This seems to go counter to the observation made earlier that fragments, in general, should be allocated to the nodes where they are accessed. The answer to this riddle is in table 4 which shows the number of seconds of processor time utilized at each node.

In table 4 we can see that node 0 has very low utilization in all cases except the T3 with no consideration of latency. Node 1 on the other hand has very high utilization, although it is considerably lower in the T3 without latency case. Fragment 0 was consistently allocated only to node 0, except in the T3 without latency case. The reason is that fragment 0 is involved in a join with fragment 4, which is allocated to node 0. By allocating fragment 0 to that node as well, the join may be performed there, thus taking advantage of some spare processor cycles at node 0. Essentially, the failure to include latency in the consideration is causing the model to attempt to load-balance across the network. Load-balancing can be expensive in terms of the number of messages required for updating replicated fragments. By allocating fragment 0

Table 4
Processor time utilized by node.

	56K		T1		T3	
	No latency	Latency	No latency	Latency	No latency	Latency
Node 0	98	98	98	98	6,591	24
Node 1	37,840	37,840	37,840	37,881	31,110	38,004
Node 2	12,234	12,234	12,234	12,210	12,234	12,210

to both nodes 0 and 1, the number of messages required to update the fragment is significantly increased. Thus, the total number of messages sent in the system in the T3 case without latency is 403,790. By contrast, the number of messages sent when latency is considered is only 239,270. If we do not consider latency, each message is considered to be very inexpensive, and we can afford to send a large number of small messages. By contrast, when we include latency in the model, we realize that small messages incur almost the same delay as large ones. The optimal solution will then be one with fewer, but more voluminous, messages. The total communications volume, therefore, remained relatively constant at 1,379,885 and 1,310,495 kilo bytes, respectively. This indicates that in a faster network, there is a bias towards sending fewer messages containing relatively more data to take advantage of the proportionally high fixed cost of latency.

8. Future research

It is important to continue to investigate the network response time model used for distributed database design. A more elaborate version of the model shown here is under development. That model also considers issues such as parallelism between queries. In addition, such a model needs to be adapted to the types of database systems used in corporate information systems today. To that end, we are developing a program which uses a genetic algorithm approach, much like that used in [9,10], to perform distributed database design on more realistic sized problems. Lastly, such a model may also be used to design other portions of distributed information systems, such as application services.

References

[1] J.C. Brancheau, B.D. Janz and J.C. Wetherbe, Key issues in Information Systems Management: 1994–95 SIM Delphi Results, MIS Quarterly 20 (1996) 225–242.

[2] M. Broadbent, P. Weill, T. O'Brien and B.S. Neo, Firm context and patterns of IT infrastructure capability, in: *The 17th International Conference on Information Systems*, Cleveland, OH (1996).

[3] K.P. Eswaran, Placement of records in a file and file allocation in a computer network, in: *Proceedings of the IFIP Conference on Information Processing '74*, New York (1974).

[4] W.R. Stevens, *TCP/IP Illustrated* (Addison-Wesley, 1994).

[5] W.W. Chu, Optimal file allocation in a multiple computer system, IEEE Transactions on Computers C-18 (1969) 885–889.

[6] R.G. Casey, Allocation of copies of a file in an information network, in: *The AFIPS Conference Proceedings*, Vol. 40, The 1972 Spring Joint Computer Conference, Atlantic City, NJ, USA; 16–18 May 1972 (1972).

[7] R. Blankinship, A.R. Hevner and S.B. Yao, An iterative method for distributed database design, in: *Proceedings of the 17th International Conference on Very Large Databases*, Barcelona, Spain (1991).

[8] S.T. March and S. Rho, Allocating data and operations to nodes in a distributed database design, IEEE Transactions on Knowledge and Data Engineering 7 (1995) 305–317.

[9] S. Rho, Distributed database design: Allocation of data and operations to nodes in distributed database systems, in: *Information and Decision Sciences* (Minneapolis, MN, University of Minnesota, 1995).

[10] S. Rho and S.T. March, Designing distributed database systems for efficient operation, in: *Proceeding of the International Conference on Information Systems*, Minneapolis, MN (1995).

[11] S. Ram and S. Narasimhan, Database allocation in a distributed environment: Incorporating a concurrency control mechanism and queuing costs, Management Science 40 (1994) 969–983.

[12] C.T. Yu, K.C. Guh, D. Brill and A.L.P. Chen, Partition strategy for distributed query processing in fast local networks, IEEE Transactions on Software Engineering 15 (1989) 780–793.

[13] S.Y.W. Su, A microcomputer network system for distributed relational databases: Design, implementation and analysis, Journal of Telecommunication Networks 2 (1983) 307–334.

[14] A. Brunstrom, S.T. Leutenegger and R. Simha, Experimental evaluation of dynamic data allocation strategies in a distributed database and changing workloads, in: *Proceedings of the 1995 ACM CIKM International Conference on Information and Knowledge Management*, Baltimore, MD, USA, 28 Nov.–2 Dec. 1995 (1995).

[15] P.M.G. Apers, Data allocation in distributed database systems, ACM Transactions on Database Systems 13 (1988) 263–304.

[16] P.M.G. Apers, A.R. Hevner and S.B. Yao, Optimization algorithms for distributed queries, IEEE Transactions on Software Engineering SE-9 (1983) 57–68.

[17] D.W. Cornell and P.S. Yu, On optimal site assignment for relations in the distributed database environment, IEEE Transactions on Software Engineering 15 (1989) 1004–1009.

[18] D.W. Cornell and P.S. Yu, Site assignment for relations and join operations in the distributed transaction processing environment, in: *Proceedings Fourth International Conference on Data Engineering* (Cat. No. 88CH2550-2), Los Angeles, CA, USA, 1–5 Feb. 1988 (1988).

[19] B. Gavish and H. Pirkul, Distributed computer system design for large decentralized organizations, in: *Proceedings of the Seventh International Conference on Information Systems*, San Diego, CA, USA, 15–17 Dec. 1986 (1986).

[20] X. Lin, M. Orlowska and Y. Zhang, On data allocation with minimum overall communication costs in distributed database design, in: *Proceedings ICCI '93. Fifth International Conference on Computing and Information*, Sudbury, Ont., Canada, 27–29 May 1993 (1993).

[21] S. Ram and S. Narasimhan, Allocation of databases in a distributed database system, in: *Proceedings International Conference on Information Systems*, Copenhagen, Denmark (1990).

[22] A.R. Hevner and S.B. Yao, Query processing in distributed database systems, IEEE Transactions on Software Engineering SE-5 (1979).

[23] R.T. Braden, RFC 1379: Extending TCP for transactions – concepts, Internet Engineering Task Force (1992).

[24] W. Stallings and R. Van Slyke, *Business Data Communication* (Prentice-Hall, Inc., 1998).

[25] W.S. Moon and Y.S. Hong, An adaptive dynamic file migration algorithm in distributed computer systems, in: *Proceedings of 1994 IEEE Region 10's Ninth Annual International Conference. Theme: Frontiers of Computer Technology*, Singapore, 22–26 Aug. 1994 (1994).