Distributed Selective Dissemination of Information

Tak W. Yan Hector Garcia-Molina
Department of Computer Science
Stanford University
Stanford, CA 94305
{tyan, hector}@cs.stanford.edu

Abstract

To help users cope with information overload, Selective Dissemination of Information (SDI) will increasingly become an important tool in wide area information systems. In an SDI service, users post their long term queries, called profiles, at some SDI servers and continuously receive new, filtered documents. To scale up with the volume of information and the size of user population, we need a distributed SDI service with multiple servers.

In this paper we first address the key problem of how to replicate and distribute profiles and documents among SDI servers. We draw a parallel between distributed SDI and the well-studied replica control problem, adapt quorum-based protocols for use in distributed SDI, and compare the performances of the different protocols. Next we address another important problem, that of efficient document delivery mechanisms. We present and evaluate a practical scheme, called profile grouping, which exploits the geographical locality of users to cut down network traffic generated by document delivery. Finally, we carry out a sensitivity analysis to determine the parameters that have critical impact on performance, and investigate strategies to cope with the scaling up of those parameters.

1 Introduction

The exploding volume and diversity of digital information pose challenging issues in large-scale wide-area information systems. In this dynamic environment it is difficult for the user, equipped with only conventional search capability, to keep up with the fast pace of information generation. Instead of making the user go after the information, information can also selectively flow to the interested users. Traditionally, libraries and databanks provide such kind of *information filtering* service, named *Selective Dissemination of Information (SDI)* [Sal68]. A user expresses his interests in a number of long-term, continuously evaluated queries, called *profiles*. He or she will then passively receive documents filtered according to the profiles. Such SDI service will become increasingly important and form an indispensable tool for global information systems.

Much recent research (e.g, see [LT92]) has focused on providing more fine-grained and effective filtering, using relational, rule-based, information retrieval (IR), and artificial intelligence approaches. Yet little has been done on the efficiency aspect, which is also critical as SDI is going to be used on a large-scale. In [YGM94b, YGM94c] we studied the use of *profile indexes* to streamline filtering at a central site, assuming IR filtering techniques. However, to really scale up we need a distributed SDI service and the efficiency of such a service is the central topic of this paper.

Distributed SDI is a problem of distributed matchmaking. On the one hand, we have information providers that look for interested users. On the other hand, we have users who seek relevant information. What forms an efficient and reliable way to perform the matching and establish the flow from the providers to the users? One naive way to achieve this is to have the users post their profiles at each and every source that may generate useful information. The sheer number of sources renders this scheme inhibitly expensive. Further, it would be very difficult to locate all potentially relevant sources. The other extreme is to have the providers send information to each and every user, and have the users screen out irrelevant information. Again, this is clearly not scalable. Valuable network bandwidth is wasted to transmit mostly irrelevant information and a lot of wasteful local processing is done. A feasible solution is to have some intermediate third-party *SDI servers*. Such servers, well-known to both providers and users, accept profiles from the users, collect documents and match them against user profiles, and relay relevant information to users.

In this paper we address the key problem of how to replicate and distribute profiles and documents among SDI servers. To illustrate, suppose a document is sent to every server, and a profile is posted at only one server. The number of profiles at each server is low, but the document arrival rate is high. Further, the availability of the service is low, since if a server goes down, the users it serves miss documents. The opposite way is to replicate a profile at all servers, and send a document to any one of them. This way, availability is high – if a server goes down, documents can be rerouted to any other server. However, the number of profiles is high at each server, and the cost of updating a profile is high also.

There are intermediate solutions in between these two. If we denote the set of servers that a profile x is posted at by P_x, and the set of servers that a document y is sent to by D_y, then to ensure that a profile does not miss a document, we must guarantee that P_x intersects D_y for every x and y. This parallels the idea of *quorum consensus* in replicated data management. In this paper, we formalize the correspondence and, given the options for replication and distribution, we study the tradeoff between communication costs, document delivery times, reliability, and other parameters. As we will see, even though the problem of distributed SDI is conceptually similar to that of quorum consensus, the different semantics of the problem and the different parameter values lead to very different conclusions regarding good strategies.

Once a document is matched to a set of profiles at an SDI server, there is the additional problem of sending the document to the users that posted the profiles. In general, there could be a very large number of users that need to receive the document. Without efficient document distribution mechanisms, much wide-area network (WAN) traffic would be generated. (The problem is related to message broadcast, e.g., [DGH+87].) To illustrate, say we have a collection of users at an institution that have posted profiles. A straightforward strategy for an SDI server is to send matching documents directly to users. The disadvantage of this approach is that if a given document happens to be of interest to many users at the institution, many copies will be sent over the WAN. An improvement may be to have a *local distribution site* at the institution. Now, if a server discovers that there is a document that matches any profile from that institution, only one copy needs to be sent to the distribution site, which then distributes the document to the relevant user(s) locally. We call this idea of viewing a group of profiles as a single delivery unit *profile grouping*. Our assumption here is that the local distribution can be done without going through the WAN. This way, WAN congestion is reduced, possibly resulting in faster document delivery. On the other hand, it makes the delivery mechanism a bit more complex. In this paper we investigate whether profile grouping provides sufficient savings to be worthwhile.

Incidentally, we have implemented at Stanford two SDI servers, one for disseminating Netnews articles and the other for Computer Science Technical Reports.[1] Recently, we publicized the Netnews server in two newsgroups; within ten days of the announcement, we received well over a thousand profiles, submitted by users from almost every continent. The number of profiles keeps increasing and now (July 1994) exceeds seven thousand. Clearly, a centralized server does not scale with the number of users (or profiles) and provides low availability. We intend to distribute the service, and this motivates the work reported in this paper. We also make use of real statistics collected from the running Netnews server to determine the values of parameters in the performance evaluation model in this paper.

[1]The reader is encouraged to try out the services. For email access, please send an electronic message to `netnews@db.stanford.edu` or `elib@db.stanford.edu`, with the word "help" in the body. For World-Wide Web access, please connect to `http://woodstock.stanford.edu:2000`.

2 Quorums for Distributed SDI

Replica control is a well-studied problem. Suppose a data item x has a number of copies. If we execute a read (write) operation on x by accessing copies that make up a read (write) quorum, then to guarantee one-copy equivalence, we must enforce the following [AA90a]:

- **Write-write Intersection Property:** Two write quorums W and U must have a non-empty intersection: $W \cap U \neq \emptyset$.
- **Read-write Intersection Property:** A write quorum W and a read quorum R must have a non-empty intersection: $W \cap R \neq \emptyset$.

In distributed SDI, we have multiple servers available. A document may be sent to more than one servers, and a profile may be posted at more than one servers. If we call the set of servers that a document is sent to a *document quorum*, and the set of servers that a profile is posted at a *profile quorum*, then to guarantee that a profile does not miss a document, the following must be satisfied:

- **Profile-document Intersection Property:** A profile quorum P and a document quorum D must have a non-empty intersection: $P \cap D \neq \emptyset$.

As we can see, the problem is analogous to replica control. In the rest of this section we briefly list the main quorum-based replica control protocols and show how to adapt them for SDI.

2.1 Majority Consensus Protocol

Majority consensus is proposed by Thomas [Tho79]. In the distributed SDI scenario, suppose we have n SDI servers. A document is sent to a document quorum formed by d (arbitrary) servers. A profile is posted at a profile quorum formed by p (arbitrary) servers. If we enforce the equality

$$p + d = n + 1,$$

then the profile-document intersection property is guaranteed; i.e., for every profile-document pair, there must be at least one SDI server that manages the profile and also receives the document. Note that to update a profile (including submission, modification/feedback, and deletion), p servers have to be accessed.

As an example, suppose we have eight SDI servers as in Figure 1(A), with $p = 3$ and $d = 6$. A particular profile is posted at the three arbitrary servers shown, and a given document is being sent to six arbitrary servers. We can see that for the particular document-profile pair shown, there are two servers in the intersection between the profile and document quorums. In this case, the matching for this profile-document pair will be performed by both servers. We assume that duplicate documents are screened out at the user site. Later in Section 5 we will see that such extra matching work and duplicate delivery lead to poor performances. We denote a majority consensus arrangement as M(p, d); thus the organization in Figure 1(A) is M(3, 6).

2.2 Grid Protocol

Cheung, Ammar, and Ahamad [CAA90] propose the grid protocol for managing replicated data. Here we adapt the protocol for use in distributed SDI.

In the grid organization, we divide the servers into d rows, each having p servers. A profile quorum is formed by selecting a row at random (p servers), and a document quorum is formed by selecting a (random) representative from every row (d servers). We denote such an arrangement as G(p, d). For instance, in Figure 1(B), eight SDI servers are arranged into a 2×4 grid. A particular profile is shown to be posted at the servers in one of the two rows, and a given document is being sent to an arbitrary server in each of the rows. Note that in the grid protocol, there is always only one server in the intersection between any document quorum and any profile quorum. It is this server that performs the matching for a particular profile-document pair.

2.3 Tree Protocol

The tree quorum protocol, proposed by Agrawal and El Abbadi [AA90b], may also be extended to the SDI scenario. However, in this protocol, the servers would be organized into a logical tree, and the server at the root would be responsible for a heavier load (more profiles and documents) than the others. This may be useful in certain circumstances, but we are more interested in the scenario in which all SDI servers share balanced load. Thus we do not consider this protocol in our performance evaluation.

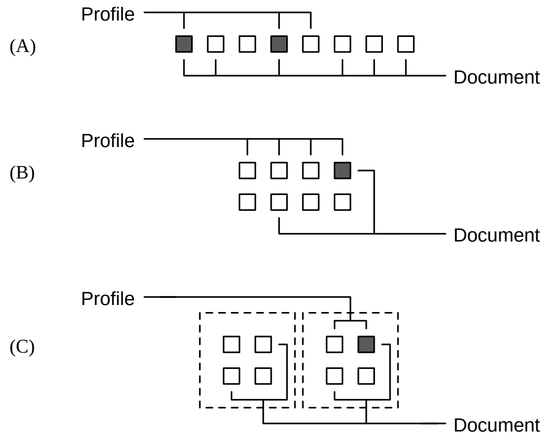

Figure 1: Quorum Protocols for Distributed SDI

2.4 Hierarchical Protocols

To trade off data availability with communication costs, researchers have proposed to compose the basic quorum protocols for replicated data. The physical copies of a data item are divided into logical units. Within a unit, the copies are organized using a certain quorum protocol. Then, each unit is considered as a logical copy of the data item and all units are in turn arranged into logical units organized by a (perhaps different) quorum protocol.

These proposals include some two-level proposals, such as grid-set [AA90a] (in which the grid protocol is used at the physical level and the majority consensus protocol is used at the logical level) and RST [RST92] (in which majority consensus is used at the physical level and the logical level is a grid). There are also ones with an arbitrary number of levels, such as hierarchical consensus [Kum91] (majority consensus used at all levels) and hierarchical grid [KC91] (grid protocol at all levels).

Each of these can be adapted to distributed SDI. To illustrate, consider Figure 1(C). We have again eight SDI servers. Suppose we first split the servers into two logical units, each with four servers. Within each unit, we choose to arrange the servers into a 2×2 grid. Next we view the two units as two logical servers, and coordinate them using the majority consensus protocol. To post a profile, we select one of the units. Within that unit, we then post the profile at the servers in one of the rows. A document is sent to both units, and a server from each row is selected to receive the document.

In addition to these proposals, we may of course have other arbitrary combinations; e.g., each logical unit could be coordinated using a different protocol. For simplicity we limit ourselves to an evaluation of two-level hierarchies made up of the majority consensus and grid protocols only. This gives us the following four arrangements:

- grid-grid (GG) – grid protocol used at both physical and logical levels;

- grid-majority (GM) – grid at physical level and majority consensus at logical level;
- majority-grid (MG) – majority consensus at physical level and grid at logical level;
- majority-majority (MM) – majority consensus at both physical and logical levels.

We extend our notation to these hierarchical quorums. For instance, for the grid-majority configuration with physical level profile quorum size p_0 and document quorum size d_0, and logical level profile quorum size p_1 and document quorum size d_1, we write GM(p_0, d_0, p_1, d_1). (The one in Figure 1(C) is GM(2, 2, 1, 2).) We may also use $\mathrm{GM}_n(p_0, d_0)$ to refer to the different possible configurations that have physical level profile quorum size p_0 and document quorum size d_0 for arranging n servers. In context where n is unambiguous, we may drop the subscript. We adopt similar conventions for other arrangements.

3 Profile Grouping

After an SDI server determines the set of profiles that match an incoming document, it may directly send the document to the individual users that have posted those profiles. An alternative to this, called profile grouping (see Section 1), may reduce communication costs by distributing a single copy of a document to each cluster of geographically local users.

More generally, assume that users are partitioned logically into user groups. Communication within the group is inexpensive, without going through the WAN. Suppose that users in a group all post their individual profiles at the same profile quorum; profiles coming from a user group form a profile group. An SDI server processes incoming documents against each individual profile just as before. However, after it determines that a document is relevant to at least one profile in a profile group, it sends out a single copy of the document to some local distribution site associated with that group. The copy of document sent to the group contains a control header that specifies which users in the group are to receive a copy of the document. The document is then distributed locally to the appropriate users.

The grouping of profiles is logical. Profiles from users on a local area network (LAN) can form a group, or a LAN can have multiple groups to keep the group size small and the local distribution fast. The criteria is that the communication costs within the group are negligible compared with the costs communicating through the WAN.

The local distribution site runs a simple program to distribute the document according to the control header. This local distribution process may add to the delay in document delivery, but the reduction in network congestion may result in faster WAN transmission that outweighs the former. It may also reduce the availability of the system, since if the local distribution site goes down, the users in the group miss documents. However, we may assume that the local distribution service, which is very simple, is replicated enough times to guarantee high availability.

Profile grouping provides greater savings if documents frequently match many profiles in a group. We expect this to be indeed the case because we believe users usually share some common interests. In Section 4.1 we develop a model to capture the similarity between profiles. In Section 4.2 we present evidence from our Netnews SDI server to validate the claim.

4 Performance Evaluation

To compare the various quorum organizations and to determine whether profile grouping is worthwhile, we carry out an analytical performance evaluation. While the presented strategies apply to any filtering techniques, for concreteness in evaluation we assume a well-known IR approach, the *vector space model (VSM)* [Sal89]. The WAIS [KM91] system uses this model, as do our experimental SDI servers (see end of Section 1). Briefly, VSM represents documents and profiles as vectors of weighted terms; e.g., a sample profile would be ⟨(graphical, 0.75), (user, 0.30), (interface, 0.60)⟩. A document matches a profile if the *cosine similarity measure* between their vector representations is higher than a user-specified threshold. We further assume that an SDI server makes use of a profile index [YGM94b] (indexing a profile by its terms) to speed up the matching. These assumptions are necessary for modeling the filtering process done at an SDI server.

Below we first describe our analytical model and explain the base values we select for the parameters. We then present the metrics used in the evaluation. Due to space considerations, we omit the details of the analysis, which can be found in [YGM94a].

4.1 Analytical Model

We assume that the combined document generation rate from all information sources follows a Poisson distribution, with an average of λ document/sec. The document size follows an exponential distribution with mean s_d words.

We assume the total number of profiles is m. The profile update (including submission, modification/feedback, and deletion) rate follows a Poisson distribution, with mean proportional to the the number of profiles. Parameter ν (update/profile/sec.) is the proportionality constant. We assume the size of a update message is exponentially distributed, with mean s_p bytes.

There are n SDI servers available. A profile is posted (or replicated) at p servers, $1 \leq p \leq n$, and a document is sent to d SDI servers, $1 \leq d \leq n$; p and d vary for different quorum configurations. A random document-profile pair has a probability θ of being a match.

To consider resource contention, each SDI server is modeled as an $M/G/1$ server, servicing two kinds of jobs: documents to be filtered and profile updates. An incoming document is matched against an index of profiles, and the time this takes is proportional to the number of profiles at the server and the size of the document [YGM94b]. (The processing time also depends on the average profile length, but we assume this value is unchanged in this paper.) We denote the proportionality constant by c (sec./profile/word). Since at a particular server the the number of profiles is fixed (for a particular organization), and the document size is exponentially distributed, the processing time is also exponentially distributed. To process a profile update, an SDI server needs to modify the profile index. We assume this takes time exponentially distributed with mean u sec. [2]

The availability of a server, i.e., the probability that it is operational, is a_0. This includes hardware, software, and communication failures.

For the WAN, instead of modeling a particular existing network, we opt for a simple, generic topology in which a number of switching nodes are fully connected by l channels (note that if x is the number of switching nodes, $l = x(x-1)$), each of bandwidth b Kbps. This model captures communication parallelism and resource contention, and is simple enough for us to derive closed-form solutions. Again for simplicity, we assume that the users, information sources, and SDI servers are uniformly distributed across the WAN, and thus the amount of traffic through each channel is the same. Each channel is modeled as an $M/G/1$ server, processing new documents and profile update messages. As we assume that the sizes of documents and update messages follow exponential distributions, the respective transmission times also follow exponential distributions.

In modeling profile groups, we assume the group size is k. We need to determine the probability that a new document matches one or more profiles in a group. If we assume that the profiles are mutually independent, this probability is given by

$$P(\theta, k) = 1 - (1-\theta)^k.$$

However, as discussed in Section 3, we believe that it is often the case users in a group have similar interests. The independent profiles assumption overestimates the probability that a document is of interest to a group. We may instead think of a group of non-independent profiles having the same selectivity as a *smaller* group of *independent* profiles. We model this by introducing a parameter α, $0 \leq \alpha \leq 1$, called the profile coalescence factor and assume that the number of independent profiles is k^α. Using this, the probability is given by

$$P(\theta, k, \alpha) = 1 - (1-\theta)^{k^\alpha}.$$

In the following subsection we determine an empirical value for α using data from our Netnews SDI server and use it in our evaluation. We assume the local distribution time is negligible compared to the WAN transmission time and SDI server matching time. Thus we do not model the local distribution site.

4.2 Estimation of Parameter Values

Table 1 shows a summary of the parameters in our evaluation model, together with the base values. For some parameters, we make use of statistics collected from our Netnews SDI server to estimate the base values. For the rest, we choose values that we believe are reasonable and are useful for illustrating the key tradeoffs. Keep in mind that all the base values of Table 1 merely represent a reasonable starting point for our evaluation. In Section 5.3 we perform a sensitivity analysis to study the effects of changing the various parameters.

Parameter	Base Value	Description
λ	0.48	mean document generation rate (document/sec.)
s_d	323	mean document size (words)
m	10^6	total # profiles
ν	7.0×10^{-7}	proportional constant for mean profile update rate (update/sec./profile)
s_p	50	mean update message size (bytes)
n	16	# SDI servers
p	1 - 16	a profile is replicated at p servers
d	1 - 16	a document is sent to d servers
θ	$3.0\times 10^{-}4$	probability that a profile matches a document
c	10^{-8}	proportionality constant for mean filtering time (sec./profile/word)
u	10^{-2}	profile update processing time (sec.)
a_0	0.8	server availability
l	72	# links in WAN model
b	50	bandwidth of WAN link (in Kbps)
k	1,000	profile group size
α	0.92	profile coalescence factor

Table 1: Summary of Model Parameters

[2] Here we are assuming that the update processing time is not related to the update message size. This seems to be reasonable as for instance, a deletion message is short in size but may take long to process.

4.2.1 Using Operational Statistics

We compute the base values for λ, θ, ν, and α using operational data from our Netnews SDI server. Table 2 shows some statistics for the week of February 25 to March 3, 1994. These include the number of incoming documents, number of profiles, and number of updates (columns 2, 3, and 4 respectively). In column 5 we show the total count of matched documents (including duplicates), and in column 6 we show the number of distinct matched documents. A document that matches two profiles is counted twice in column 5 and only once in column 6.

Day	# arriving docs	# profiles	# updates	# matched docs	
				total	distinct
2/25	49,649	1,527	44	20,717	10,504
2/26	43,034	1,525	63	19,603	9,956
2/27	26,874	1,538	79	16,340	7,690
2/28	30,635	1,550	145	16,436	8,172
3/1	41,003	1,649	118	20,474	9,981
3/2	61,523	1,640	96	22,629	12,064
3/3	38,967	1,645	117	19,493	9,726

Table 2: Statistics Collected from Netnews SDI Server

We calculate λ as the average number of incoming documents divided by the number of seconds in a day. Parameter θ is calculated as the average total number of matched document divided by the product of the average number of profiles and the average number of incoming documents. We calculate ν as the average number of updates per day divided by the average number of profiles divided by the number of seconds in a day.

To estimate α we need a sample group of profiles. One choice is to consider all profiles at our server to be one group. Even though our users come from all over the world and they are not connected by a LAN, they have one important thing in common: they are subscribing to our particular server. If we do this, and solve the following equation for α (assuming the expression for $P(\theta, k, \alpha)$ in Section 4.1):

$$\begin{aligned}&(\text{avg. \# incoming documents}) \times \\ &\quad (1-(1-\theta)^{(\text{avg. \# profiles})^{\alpha}}) = \\ &\qquad (\text{avg. \# distinct matched documents}),\end{aligned}$$

we find α to be 0.92 (the value of θ is first obtained as described above). Note that this apparently insignificant value of α actually corresponds to quite remarkable shrinkage; for example, for 1,000 profiles, $1,000^{0.92} = 575$, a 42% decrease in the number.

Another option would be to consider subscribers from a particular institution forming a group. In this case, we found that the largest group of profiles comes from a company, made up of 56 profiles. We carried out a similar analysis, and found α to be 0.97. This larger α value means that subscribers from this company have *less* overlap in their interests than our subscribers overall. Apparently, subscribers at one institution can have very diverse interests. We conclude that a reasonable range for α is from 0.92 to 1 and use 0.92 as our base value. In Section 5.2 we determine the range of WAN traffic reduction produced by profile grouping over this range of values of α.

4.2.2 Base Values for Other Parameters

In an analysis of a 550MB database of Netnews articles reported in [YGM94b], we found the average number of words in an article to be 323. Thus we pick s_d to be 323. We use analysis and simulation results from [YGM94b] to estimate the time it takes to match a document against a profile index. We find that to match a document of 323 words against 300,000 profiles, it takes 144 I/Os and 4,313 floating-point multiplications. We thus assume that it takes 1 sec. (order of magnitude) to match, and estimate c as 10^{-8} sec./profile/word ($1/300,000/323 \approx 10^{-8}$). We assume the average size of a profile update message (s_p) is 50 bytes and estimate the mean update time u to be 0.01 sec.

Reference [LCP91] reports that the availabilities of some 68,000 Internet hosts range from 0.7833 to 0.9688. We choose 0.8 for the SDI server availability (a_0), a number on the low end to factor in the failures of document filtering and delivery software.

Reference [Rei93] estimates that as of January 1993, the Netnews readership worldwide is 1.9 million. We thus choose a base value of 1 million for m, the total number of profiles. We assume there are 16 SDI servers, and 9 switching nodes in the WAN, giving us a value of $9 \times (9-1) = 72$ for l, the number of channels. The WAN bandwidth b is taken to be 50 Kbps ([GR93] 1990 figure). Of course we expect larger bandwidth in the future, but at the same time we expect greater volume of information, larger user population, and larger documents (recall that Netnews articles average only a few hundred words per article). We believe these figures together give us a compatible starting set of values to study the tradeoff.

4.3 Evaluation Metrics

We evaluate the performances of the different strategies using three metrics. The first is the document delivery delay t, i.e., the total time elapsed between the time a document is generated and the time the interested user receives the document. This metric is relevant for applications in which the timeliness of information is critical, such as news stories, financial services, stock updates, and others. The second metric is the WAN utilization, ρ. It is important for situations in which the WAN is a valuable resource.

The last metric a measures the availability of the distributed SDI system. Here we define system failure as

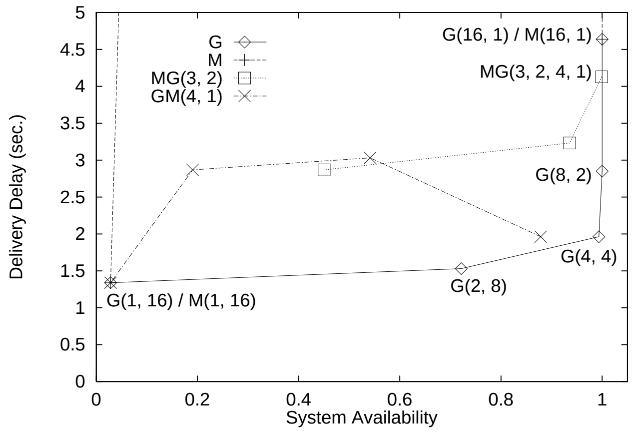

Figure 2: Document Delivery Delay vs. System Availability

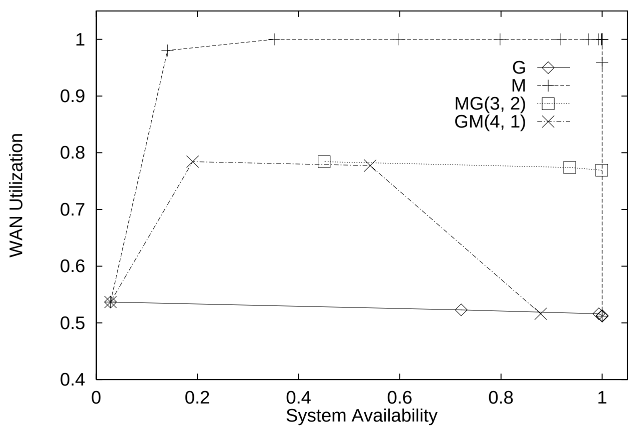

Figure 3: WAN Utilization vs. System Availability

the event that a user is missing documents, i.e., not receiving documents that match his or her profiles as soon as possible. There are other kinds of undesirable behavior, such as the event that a user cannot submit a new profile to the appropriate number of servers. However here we are more concerned with the requirement that once a user subscribes, he or she receives uninterrupted delivery of documents. Thus we define failure as above, and define system availability as the probability that there is no failure at a particular time.

5 Results

5.1 Comparing Quorum Structures

We enumerated all possible quorum configurations using the six protocols discussed in Section 2 and evaluated them according to the three metrics. To study the tradeoff between performance and availability, we plot two kinds of graphs: in the first, we show the delivery delay against the system availability, and in the other, we graph the network utilization against the system availability. The detailed graphs are presented in [YGM94a]. Here for clarity we only show the plots for the grid, majority, GM(4, 1) and MG(3, 2) strategies to illustrate the kind of results obtained and to explain the basic tradeoff that we found applies to all other organizations.

Figure 2 shows the graphs for delivery delay (y-axis) vs. system availability (x-axis). Each data point represents a particular quorum organization. For example, if we look at the graph for the grid organizations (diamond symbol), the leftmost data point is for G(1, 16), followed by those for G(2, 8), G(4, 4), G(8, 2), and G(16, 1).

Focusing on the graph for the grid organizations, we see that availability increases with the profile quorum size. However, the larger number of profiles that an SDI server has to handle leads to heavier load and longer delivery delay. Thus the graph slopes upwards. If we turn our attention to the majority organizations, we observe that they perform poorly in terms of delivery delay. In fact, only two – M(1, 16) and M(16, 1) – appear in the figure (with delays less than 5 sec.). The worst ones are those with simultaneously large profile and document quorums, resulting in saturation of SDI system components.

The best organization would be the one with the highest availability and the shortest delivery delay; i.e., as close as to the bottom-right corner of the figure as possible. There are two candidates: G(2, 8) and G(4, 4), and the latter is apparently better as it provides a high availability. The grid organizations form a delay vs. availability envelope: no organization gives superior tradeoff. And in general, grid organizations with balanced document and profile quorum sizes provide high system availabilities and short delivery delays. Some boundary cases of the majority and hierarchical organizations do perform the same as a grid organization and their data points coincide in the figure. For example, M(1, 16) is the same as G(1, 16). Some, such as MG(3, 2, 4, 1), provide additional tradeoff points on the envelope not attainable with the grid.

Figure 3 shows the tradeoff between network utilization (y-axis) and system availability (x-axis). Similarly, the best organization should be the one with the highest availability and the lowest utilization; i.e., as close as to the bottom-right corner of each figure as possible. We can see that again the grid organizations envelop the others. However, a number of quorum organizations overlap at the bottom-right corner, as the differences between the utilization levels of these organizations are very small. We can also see that for most of the majority quorum organizations, the network is utilized 100%.

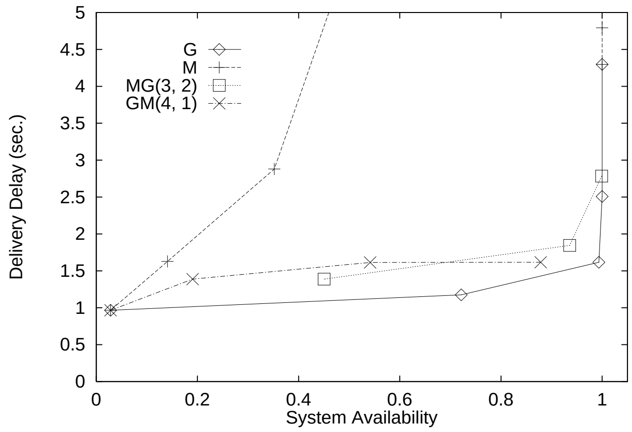

Figure 4: Document Delivery Delay vs. System Availability: Using Profile Grouping

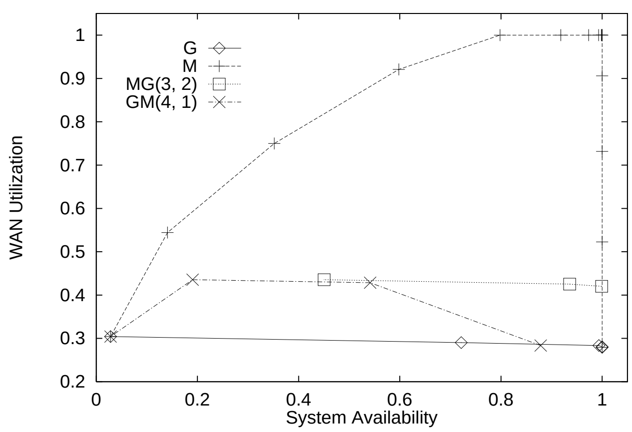

Figure 5: WAN Utilization vs. System Availability: Using Profile Grouping

5.2 Profile Grouping

In Figures 4 and 5, we show the tradeoff between delivery delay/network utilization (y-axis) and system availability (x-axis) when profile grouping (group size = 1,000, coalescence factor = 0.92) is used. For each organization the delivery delay is shorter than when no profile grouping is used, as a result of lower WAN utilization. The improvements are especially marked for the majority quorum organizations. The grid organizations still provide the tradeoff boundary.

Next we experiment with the two parameters that control profile grouping in the model. We assume two values of α: 0.92 and 1.0 and we vary the group size from 1 to 2,000. Figure 6 shows the changes in the WAN utilization for the G(4, 4) organization. As expected, the greater the group size, the greater is the reduction in WAN utilization. However, we remark that the group size is physically constrained by the requirement that

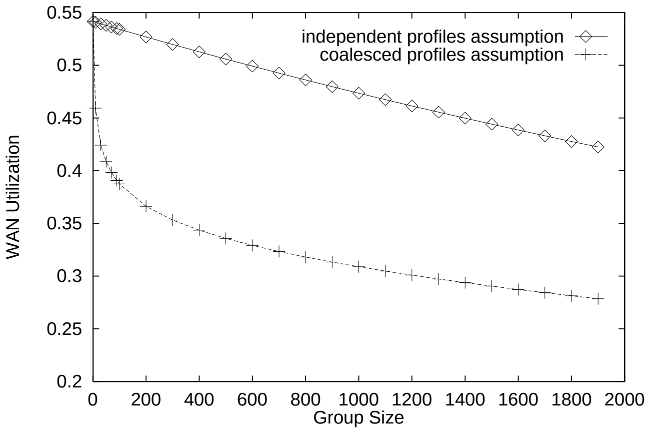

Figure 6: WAN Utilization vs. Group Size

profiles are posted by nearby users, so there is a limit to the savings profile grouping produces. Secondly, we note that the smaller α is, the greater is the decrease in WAN utilization. And even if $\alpha = 1$, i.e., assuming all profiles are independent, profile grouping is still beneficial. We also show in [YGM94a] that profile grouping is useful in controlling performance degradation when some parameters, e.g., matching probability, are increased.

5.3 Sensitivity Analysis

Here we perform a sensitivity analysis to look at how the document delivery delay is affected by varying the different parameters. We focus on the G(4, 4) organization, found to be very desirable (high availability and shortest delivery delay). We assume no profile grouping is used. For each parameter, we vary it from 60% to 250% its base value. (If x is the fraction shown on the x-axis of Figure 7, the parameter value is $v \times (1+x)$, where v is the base value.) We calculate the fractional change in the document delivery delay against the base case result. We studied all parameters, but Figure 7 only shows the results for parameters that have substantial impact on performance. Omitted are the profile update parameters, which as expected are not very critical – in an SDI service, profiles represent long term interests and profile update rate, as indicated by the numbers in Table 2, is relatively low.

The sensitive parameters include the document size, number of profiles, document arrival rate, and profile-document matching probability. The most critical one is the document size – it affects not only network traffic, but also the document filtering time.

As these parameters grow, the delivery document delay grows also. How do we cope with this in a system? For instance, in the near future, we expect very large, e.g., multimedia, documents. If we have faster networks, does this "solve" the problem? That is, can we scale to "larger" scenarios just by adding bandwidth? Or do we

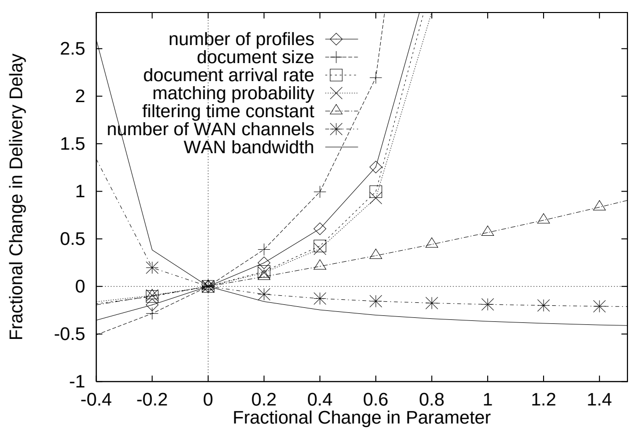

Figure 7: Sensitivity Analysis

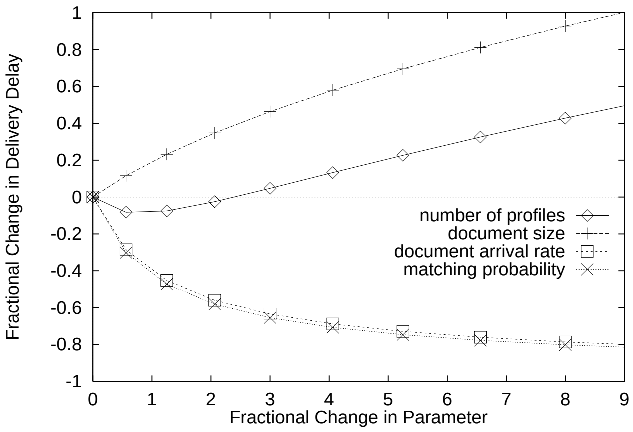

Figure 9: Increasing Bandwidth and Adding Servers

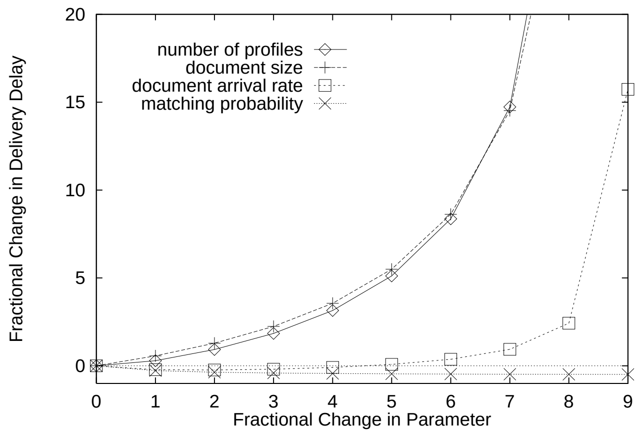

Figure 8: Increasing WAN Bandwidth

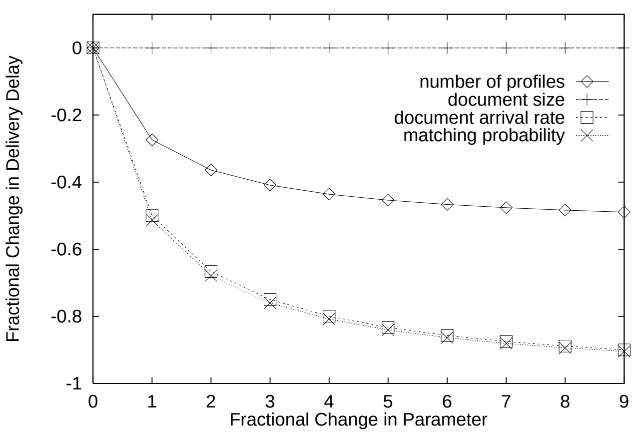

Figure 10: Increasing Bandwidth and Decreasing Filtering Constant

also need more SDI servers? To answer these questions, we carry out a number of experiments [YGM94a]. In each experiment, we try a different strategy to control the growth in the delivery delay as the parameters are scaled up. Due to space constraints, we only show the results for three interesting cases below.

Increasing Network Bandwidth First we examine if we can cope with the scaling-up of the parameters by increasing the network bandwidth alone. While increasing each studied parameter, we increase the network bandwidth to keep the ratio (parameter value)/(network bandwidth) constant. For instance, for the last data point (fractional change = 9) for document arrival rate (box symbol) in Figure 8, we increase the number of profiles and network bandwidth ten times simultaneously. The results show that increasing bandwidth effectively controls the increase in profile-document matching probability, and to a large extent the document arrival rate. However, when the number of profiles or document size is increased, or when the document arrival rate is very high (e.g., eight times the base value), the SDI servers become the bottleneck of the system.

Increasing Network Bandwidth and Adding SDI Servers Again we increase each parameter in turn, and at the same time, increase the network bandwidth and the number of servers proportionally. We achieve the latter by maintaining a constant ratio between the total number of servers and the studied parameter as in the base case. For example, for the second data point (fractional change = 0.5625) in the graph for document size ("+" symbol) in Figure 9, we assume we have 25 servers and use the G(5, 5) configuration. The fractional increase in the number of servers is (25-16)/16 = 56.25%, and thus we increase the document size (and network bandwidth) by 56.25% also. The rest of the data points in that graph correspond to G(6, 6), G(7, 7), and so on. From the figure, we can see that by increasing the bandwidth and the number of servers together, we can maintain good performance to a large extent. That is, the system can cope with larger documents or more profiles, as long as the bandwidth and the number of servers increase accordingly.

Increasing Network Bandwidth and Reducing Filtering Proportionality Constant Finally, we repeat the procedure, but this time we proportionally increase the WAN bandwidth and *decrease* the proportionality constant for the mean filtering time. The latter controls the time it takes for a SDI server to filter a document. We find that performance improves for *all* studied parameters (Figure 10), indicating that fast filtering at a SDI server is very effective, more so than adding servers, in coping with large scale scenarios. More sophisticated profile indexing schemes, such as those proposed in [YGM94b, YGM94c], are attractive. Hardware solutions (e.g., using parallel computers and RAIDs) to reduce the processing time may also be considered.

6 Conclusion

In this paper, we investigated practical strategies in the design of a distributed Selective Dissemination of Information (SDI) service. We find that a good document and profile distribution scheme, such as the grid protocol with balanced profile and document quorum sizes, is essential for providing efficient and highly-available service. Efficient document delivery mechanism, such as profile grouping, is useful in reducing network utilization and thus delivery delay. To scale up with the increase in parameters such as document size, information generation rate, and number of profiles, greater network bandwidth is required. This is not surprising, but greater bandwidth alone will not be enough. We need more SDI servers, organized in a good distribution scheme. We also find efficient index support for the filtering process performed at a SDI server to be critical.

References

[AA90a] D. Agrawal and A. El Abbadi. Exploiting logical structures in replicated databases. *Information Processing Letters*, 33:255–60, 1990.

[AA90b] D. Agrawal and A. El Abbadi. The tree quorum protocol: An efficient approach for managing replicated data. In *Proc. VLDB Conference*, pages 243–54, 1990.

[CAA90] S.Y. Cheung, M.H. Ammar, and M. Ahamad. The grid protocol: A high performance scheme for maintaining replicated data. In *Proc. International Conference on Data Engineering*, pages 438–45, 1990.

[DGH+87] A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, and D. Terry. Epidemic algorithms for replicated database maintenance. In *Principles of Distributed Computing*, 1987.

[GR93] J. Gray and A. Reuter. *Transaction Processing: Concepts and Techniques.* Morgan Kaufmann, San Mateo, California, 1993.

[KC91] A. Kumar and S.Y. Cheung. A high availability $\sqrt{n}$ hierarchical grid algorithm for replicated data. *Information Processing Letters*, 40:311–6, 1991.

[KM91] B. Kahle and A. Medlar. An information system for corporate users: Wide Area Information Servers. *Connexions – The Interoperability Report*, 5(11):2–9, 1991.

[Kum91] A. Kumar. Hierarchical quorum consensus: a new algorithm for managing replicated data. *IEEE Transactions on Computers*, 40(9):996–1004, 1991.

[LCP91] D.D.E. Long, J.L. Carroll, and C.J. Park. A study of the reliability of internet sites. In *Proc. IEEE Symposium on Reliable Distributed Systems*, pages 177–86, 1991.

[LT92] S. Loeb and D. Terry. Information filtering. *Communication of the ACM*, 35(12):26–81, 1992.

[Rei93] B. Reid. USENET Readership summary report for January 1993. *USENET Newsgroup news.lists*, Feb 8 1993.

[RST92] S. Rangarajan, S. Setia, and S.K. Tripathi. Fault tolerant algorithms for replicated data management. In *Proc. International Conference on Data Engineering*, pages 230–7, 1992.

[Sal68] G. Salton. *Automatic Information Organization and Retrieval.* McGraw-Hill, New York, 1968.

[Sal89] G. Salton. *Automatic Text Processing.* Addison Wesley, Reading, Massachusetts, 1989.

[Tho79] R.H. Thomas. A majority consensus approach to concurrency control for multiple copy databases. *ACM Transactions on Database Systems*, 4(2):180–209, 1979.

[YGM94a] T.W. Yan and H. Garcia-Molina. Distributed selective dissemination of information. Technical Report (in preparation), Stanford University, 1994.

[YGM94b] T.W. Yan and H. Garcia-Molina. Index structures for information filtering under the vector space model. In *Proc. International Conference on Data Engineering*, pages 337–47, 1994.

[YGM94c] T.W. Yan and H. Garcia-Molina. Index structures for selective dissemination of information under the boolean model. *ACM Transactions on Database Systems*, scheduled to appear June 1994.