Multimedia Systems (1996) 4: 131–139

Multimedia Systems
© Springer-Verlag 1996

A hierarchical architecture for buffer management in integrated services networks

Wei Yen*, Ian F. Akyildiz**

Broadband and Wireless Networking Laboratory, School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30332, USA

Abstract. Due to their large bandwidth demand and synchronization requirements, multimedia applications, in general, consume buffers of huge size, which prevents potential customers from using multimedia services. We recognize the problem and propose a hierarchical architecture to reduce the buffer size. The architecture can be applied to both 1-n and n-n applications. We establish the architecture by first determining neighbor sets and then applying a grouping algorithm and a renegotiation process. This architecture can also meet the synchronization requirements of multimedia applications. We evaluate the performance of the architecture through simulations and compare it with that of a *direct* connection architecture. The result shows that the hierarchical architecture reduces the buffer size significantly without serious penalty to the total bandwidth and without introducing extra hot spots.

Key words: Hierarchical architecture – Grouping policies – Renegotiation process – Prefetched buffer – Synchronization - Performance evaluation

1 Introduction

The progress of fiber optics, storage, and network technologies will make it possible to support the integration of services and to implement multimedia applications on computer networks in the foreseeable future. Asynchronous transfer-mode (ATM) technology is the widely accepted platform for future integrated services networks. Due to the nature of asynchronous transmission, the temporal relationships among media tend to be disrupted in ATM networks. To maintain the temporal relationships among media, the use of buffers at the sending/receiving sites or in the network is an inevitable consequence. The total buffer size increases with the total number of connections to users of applications, the variation of propagation delay, and the delay jitter of the connections. If the connection pattern among users is not carefully arranged, then the total buffer size will be so large that the price of running multimedia applications will become too high to be feasible.

In multimedia applications, each user is either a sender, a receiver, or both. A sender collects media units and sends them to receivers. The media units are collected from either media recorders (such as cameras and microphones), or retrieved from databases. A receiver gets all the temporally related media units from senders and plays them on display devices. A '*direct*' way to connect users in one application is to build a virtual connection between each sender and receiver pair. The virtual connection could be a virtual channel (VC) or virtual path (VP) in ATM networks. A VC is a logical connection that allows a variable bit-rate, full duplex-flow exchange between two end users, while a VP is a set of VCs with the same end points. The advantages of a direct connection are easy reconfiguration and a relatively short set-up delay. However, the direct connection may cause a waste of buffers, since receivers in the same neighborhood must buffer media units separately.

Let us consider the example given in Fig. 1. A source s transmits TV-quality video to destinations d_1 and d_2 at 30 frames/s. The maximum and minimum end-to-end delay of (s, d_1) and (s, d_2) are assumed to be 200 ms and 100 ms, respectively. Each TV-quality video frame contains 0.2–1.5 Mbits of data. To compensate the delay variance, the destinations d_1 and d_2 must buffer at least 4 frames separately. In other words, d_1 and d_2 require 1.6–12 Mbits of buffer in total.

In a teleconference application, each user is fully connected with other users, as shown in Fig. 2. Specifically, if there are N users, and if a bandwidth B is required to carry media units from one user to another, then the total number of virtual connections H_d in a direct-connection architecture can be computed from:

$$H_d = \binom{N}{2} = \frac{N^2 - N}{2}. \qquad (1)$$

We assume that the virtual connection carries all media involved in the communication such as video or voice. Moreover, the total required bandwidth B_d^* is

$$B_d^* = 2BH_d = B(N^2 - N). \qquad (2)$$

* e-mail: wei@eecom.gatech.edu
** e-mail: ian@armani.gatech.edu
Correspondence to: I.F. Akyildiz

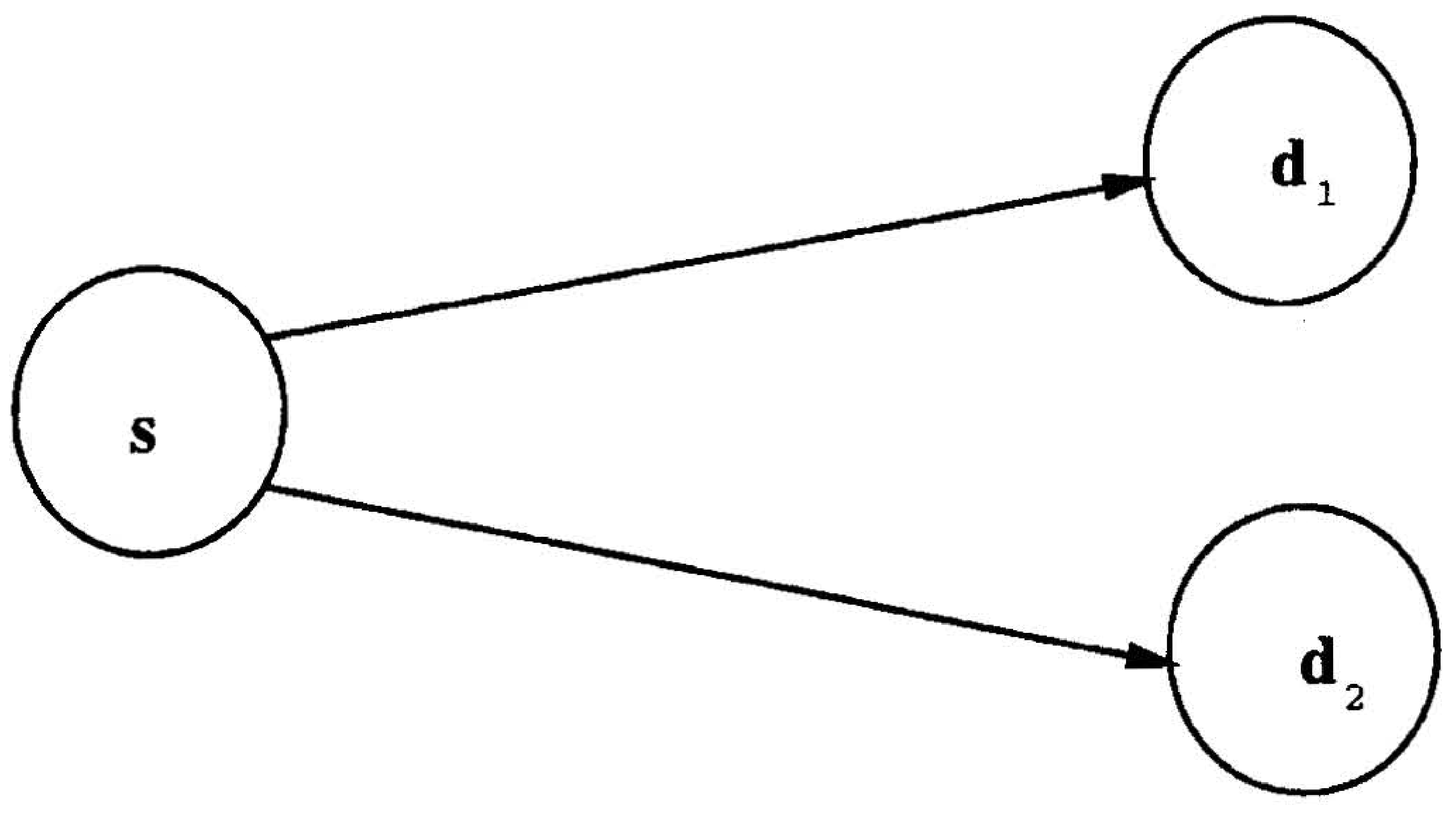


Fig. 1. Example. For both connections, the maximum end-to-end delay is 200 ms and the minimum end-to-end delay is 100 ms

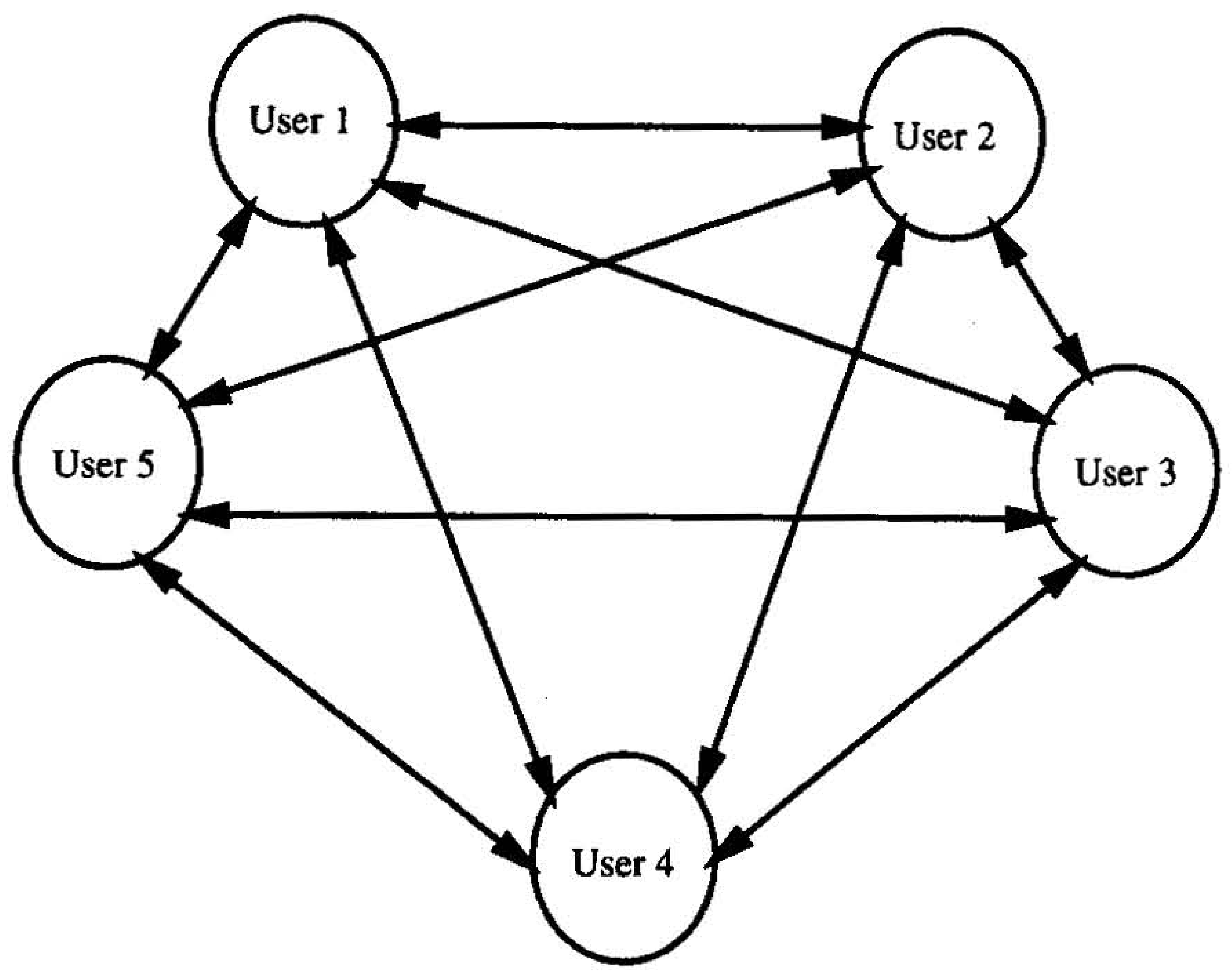
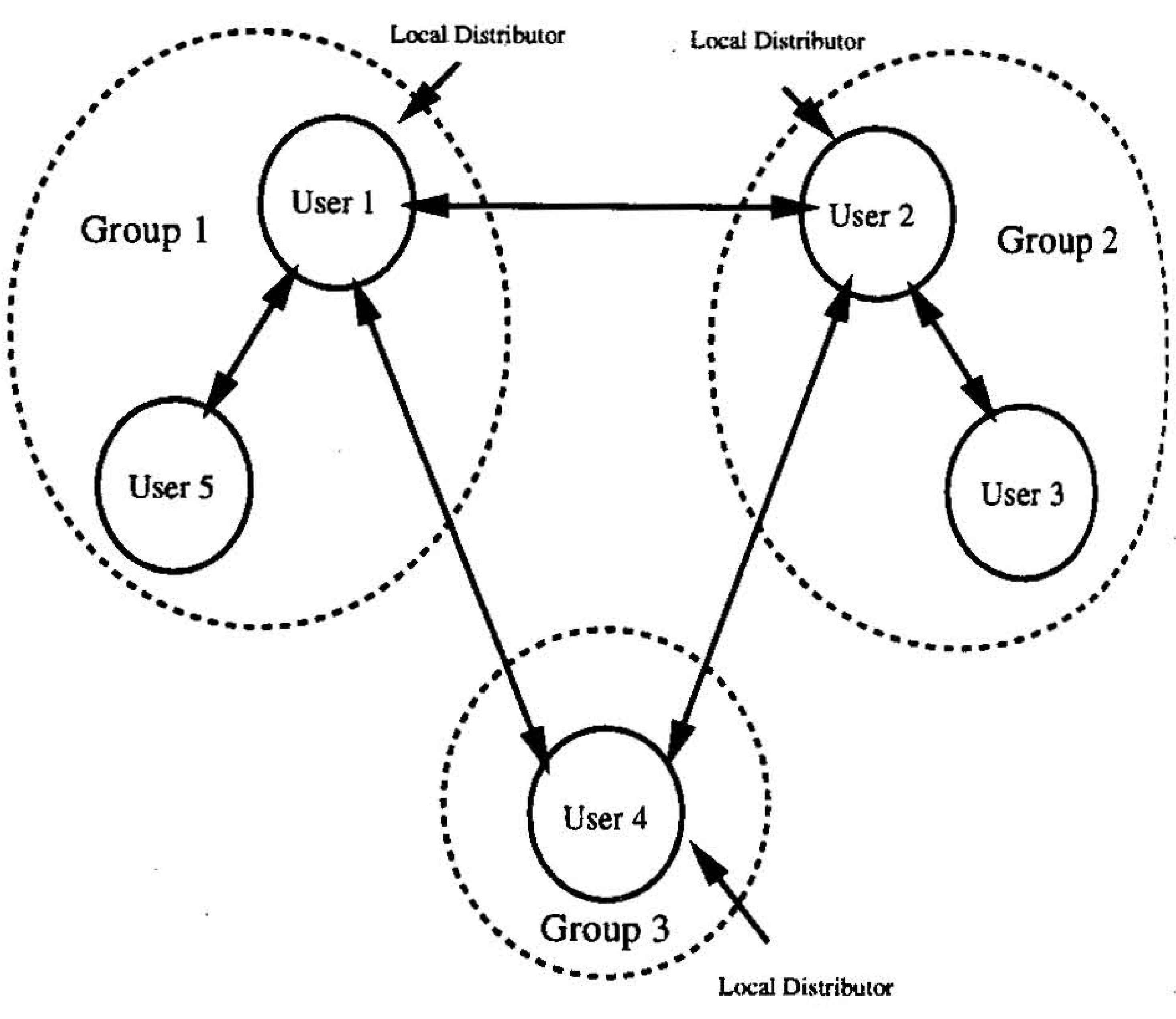
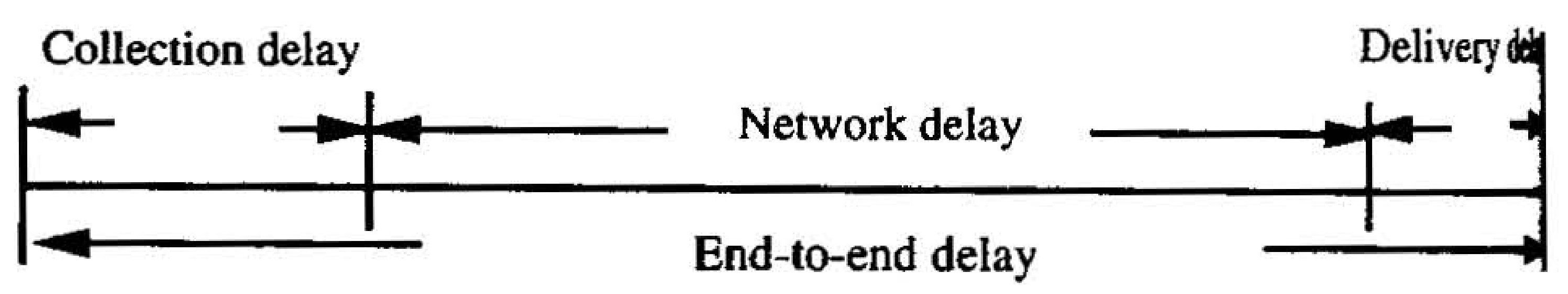


Fig. 2. Direct connection



3



4

Fig. 3. Hierarchical architecture

Fig. 4. Components of end-to-end delay

As it is clear in Eqs. 1 and 2, the total number of virtual connections H_d and the total bandwidth B_d^* increase quadratically as N increases in a direct connection architecture. For example, in Fig. 2, if $k = 1$, i.e., a single medium is communicated, and $B = 2$ Mbits/s, then $H_d = 10$ and $B_d^* = 40$ Mbits/s.

Another important issue regarding the use of a direct connection architecture is the total buffer size β required to support multimedia applications. Buffers can be at receivers, at senders, within the networks, or in a combination of these places, as required, to achieve synchronization. Wherever we put the buffers, they are one of the major cost factors for running multimedia applications. In general, the total buffer size β is a function of the end-to-end delay among users, the number of connections H, the bandwidth B, the number of users N, and the synchronization mechanism used in the applications. Our objective in this paper is to develop a hierarchical architecture to reduce the total buffer size β.

Rangan et al. (1993) propose a hierarchical communication architecture and a mixing algorithm to assure the synchronization in multimedia applications. The algorithm mixes media units in a mixer that multicasts blended media units to receivers. However, Rangan et al. (1993) have not explicitly considered the efficient use of buffers. Besides, the additional end-to-end delay is added due to the mixing algorithm. When applied in wide-area network (WAN) environments, the additional delay might be unacceptable for time-critical applications.

Adam et al. (1994) propose a software-intensive, ATM-based, network architecture, *VuNet*. In VuNet, multimedia devices are taken out of the workstation and are treated as general peripherals. Workstations access the devices through the network. The paper also addresses the internetwork problems with SONET. However, no explanation is given of how to solve the synchronization problems or how to allocate network resources efficiently.

Instead of providing a multicast solution for the integrated services networks, our goal in this paper is to develop a hierarchical architecture to reduce the buffer size requirements of users. The architecture can be applied to 1-n applications (telelecture) and n-n applications (teleconference). The 1-1 applications (nonsynchronous information retrieval) is not considered in this paper. This paper is structured as follows. In Sect. 2 we introduce two grouping policies that divide the users into several groups so that the total buffer size β can be reduced. In Sect. 3 we compute users' buffer sizes in the hierarchical distribution architecture. In Sect. 4 we introduce a synchronization mechanism over the hierarchical architecture to support time-critical multimedia applications. In Sect. 5 we explain how users can enter and leave a communication group with the least effect on existing communication. In Sect. 6 we evaluate the performance of our architecture and compare it with direct connections. In Sect. 7 we conclude the paper.

2 Hierarchical architecture for multimedia distribution

We consider a network with N users who want to communicate with one another. In other words, these N users constitute a communication group. In the direct connection architecture, each user builds connections with other users in the communication group, i.e., all users in the communication group are fully connected. We denote the bandwidth as B (which is the sum of several connections carrying different media) to transmit media units between the users. Since the total number of virtual connections H_d given in Eq. 1 and the architecture given in Fig. 2 can be reduced by appropriately grouping users, we divide the N users into G subgroups, each having n_i users, for $i = 1, 2, \dots, G$, as shown in Fig. 3. We select one user from each subgroup as a local distributor. The responsibilities of the local distributor are: to receive media units from the users in its own subgroup and send them to remote local distributors and to receive media units from remote local distributors and distribute them to other users in its own subgroup. (We assume that media are mixed at all users except local distributors. This assumption allows us to find out the worst case bandwidth and the buffer requirements.) Note that all users within a subgroup are connected to their local distributor. In the hierarchical communication architecture, the total number of virtual connections H_h becomes

$$H_h = \sum_{i=1}^{G}(n_i - 1) + \binom{G}{2} = N + \frac{G^2 - 3G}{2} \tag{3}$$

and the total bandwidth B_h^* becomes

$$\begin{aligned} B_h^* &= \sum_{i=1}^{G}(n_i - 1)(B + BN) + \sum_{i=1}^{G}(Bn_i(G - 1)) \\ &= (N - G)(B + BN) + BN(G - 1) \\ &= B(N^2 - G) \end{aligned} \tag{4}$$

The first term on the right-hand side of Eq. 3 is the sum of the connections in each subgroup; the second term is the sum of connections among local distributors. Similarly, the first term on the right-hand side of Eq. 4 is the sum of bandwidth required for each subgroup. Each user requires bandwidth B to transmit its media units to the local distributor. In addition, the local ditributor requires bandwidth BN to transmit the media units from all N users to each user in its own group. (In many applications the local distributor only requires $B(N - 1)$ bandwidth for each user in its group because the media units from each user might not need to be sent back to itself.) The second term is the sum of bandwidth required among local distributors. For $G = N$, Eqs. 3 and 4 become Eqs. 1 and 2, respectively. Note that even H_h is smaller than H_d; the hierarchical architecture may not be better than the direct connection architecture in terms of the consumption of VC entries at nodes in the network.

Although grouping can help us to reduce the total number of connections H_h, we can not do this randomly. Our grouping concept puts users within the same neighborhood in the same subgroup. In such a case, the end-to-end is upper-bounded so that the local distributor of each subgroup needs to buffer less media units. To explain the idea better, we introduce the concept of *prefetch time*, denoted by $L_{i,j}$. This is the amount of time required for user j to buffer the media units from user i after i starts to collect them. In this way, the delay-jitter effect can be compensated. We use the threshold α_i of the prefetch time $L_{i,j}$ to define the so-called "neighborhood". If the prefetch time $L_{i,j}$ is less than α_i, then j is a neighbor of i. We derive the prefetch time $L_{i,j}$ in Sect. 3.1 and determine α_i in Sect. 2.1 by considering the synchronization requirements of applications and buffer sizes available to users. We obtain the maximum end-to-end delay $\Delta_{i,j}(m)$ of the connection carrying medium m from user i to j from Escobar et al. (1994):

$$\Delta_{i,j}(m) = \delta_{i,j}(m) + J_{i,j}(m) \tag{5}$$

where $\delta_{i,j}(m)$ is the minimum end-to-end delay including the propagation delay, minimum collection delay and minimum delivery delay (Fig. 4); $J_{i,j}(m)$ is the maximum end-to-end delay jitter. We note that $\delta_{i,j}$ can be obtained by taking the minimum value over several end-to-end delay measurements Bolot (1993). Further, the collection delay is the time needed for the sender to collect media units and prepare them for transmission. The delivery delay is the time needed for the receiver to process the received media units and prepare them for playback.

Note that the neighborhood relation is not transitive. For example, if user j is a neighbor of user i and user l is a neighbor of user j, then user l is not necessarily a neighbor of user i. We need an algorithm that can decide whether (i, j) or (j, l) are in the same subgroup. We propose two grouping algorithms in Sect. 2.2 to solve this conflict. We outline the basic idea behind the hierarchical distribution architecture here.

In the establishment phase of the hierarchical architecture for the communication group, each user builds connections with other users in the communication group. Each user, i, for $i = 1, 2, \dots, N$, estimates the end-to-end delay between itself and the other users, then decides on its neighbor set, S_i, which contains the neighbors of user i and user i itself. The prefetch time $L_{i,j}$ of the connection between user i and any user j from the set S_i must be less than the precomputed upper bound α_i. After determining the neighbor set S_i, user i generates a random number r_i and sends it and the set S_i to all other users in the communication group. According to neighbor sets and random numbers from all users, the distributed grouping algorithms explained in Sect. 2.2 can decide on the connection pattern among the users. The N users of the communication group negotiate with their local distributor according to the determined connection pattern. For each local distributor, we require negotiation with its group members and other local distributors.

We need to solve three problems for the hierarchical distribution architecture:

1. The determination of the neighbor sets by deriving an upper bound α_i for the prefetch time
2. The development of grouping algorithms that divide users into several subgroups based on the neighbor sets provided by the users
3. The establishment of the negotiation process for bandwidth after the grouping algorithm is complete

2.1 Determining the neighbor sets

As mentioned before, if the prefetch time $L_{i,j}$ is less than the threshold α_i, then we put user j in the neighbor set S_i. The threshold α_i is related to the buffer size γ_j available at j to support the multimedia application. In general, a large α_i increases the buffer size for users closer to i in the same neighbor set S_i because these users must buffer additional media units to synchronize their initial playback time with those users far from i, but still in the same neighbor set S_i. Alternatively, a small α_i tends to cause inefficient use of resources and make the hierarchical distribution architecture useless because the total number of subgroups G will approach N as α_i becomes small. Therefore, we need to select an appropriate α_i. Suppose user i is the local distributor, then $t_{i,j}$ is the earliest time when γ_j, the available buffer size for user j, becomes full with media units. The following equation shows the relationship between $t_{i,j}$ and γ_j:

$$\sum_{m=1}^{k}(t_{i,j} - \delta_{i,j}(m))Nb_m = \gamma_j \quad \text{for} \quad i,j = 1,2,\ldots,N \;, \tag{6}$$

where k is the number of media transmitted between each pair of users and b_m is the bandwidth required to carry medium m. Note that $\sum_m(b_m) = B$. By rewriting Eq. 6, we solve for $t_{i,j}$:

$$t_{i,j} = \frac{\gamma_j}{NB} + \frac{\sum_{m=1}^{k}\delta_{i,j}(m)b_m}{B} \tag{7}$$

The first term of the right-hand side measures the time required to fill the buffer at j, while the second term measures the time required to fill the storage inherent in network transmission lines and nodes.

The threshold α_i is then obtained from:

$$\alpha_i = \min_j\{t_{i,j}\} \tag{8}$$

where j refers to any user. This condition guarantees that every group member has enough buffer to compensate delay jitter.

In Eqs. 6–8, it is clear that if each user j informs i about its available buffer size γ_j, then user i can determine the threshold α_i. User i can also calculate the prefetch time $L_{i,j}$ as we will demonstrate in Sect. 3.1. If the prefetch time $L_{i,j}$ for user j is less than the threshold α_i, then user j is considered to be in the neighbor set S_i. That means, any user j in the neighbor set S_i can store enough media units to compensate the delay-jitter effect without flooding the buffer size γ_j. Note that the buffer size limit γ_j is machine dependent.

2.2 Grouping algorithms

After determining the neighbor set S_i, each user i generates a one-byte random number r_i and transmits it and the set S_i to all users. A distributed grouping algorithm running at each user i determines the subgroups and local distributors based on the random numbers and neighbor sets from all users. We suggest two grouping algorithms to optimize the use of various network resources. The first algorithm optimizes the number of connections by minimizing the number of subgroups G. In Eq. 3, when N is fixed, minimizing the number of subgroups G optimizes the number of connections H_h as far as the grouping algorithm is concerned. The algorithm starts to select the largest neighbor set, say S_i, to be a group and assigns user i as the local distributor of the subgroup. Therefore, users in S_i, except for user i, can be neither members of other subgroups nor local distributors. The algorithm then deletes users in S_i from the remaining neighbor sets. The neighbor sets of the users in S_i are also deleted. The algorithm repeats for the remaining neighbor sets until there is no neighbor set left. If there are several neighbor sets that all have the largest size, then the one with the largest random number r_i is selected by the algorithm as the next group. If the random numbers of the neighbor sets are the same, then each contender regenerates a random number r_i and transmits it again.

The second grouping algorithm focusses on the total bandwidth B_h^*, Eq. 4, where a larger G results in smaller total bandwidth requirement. Thus, this algorithm tries to maximize the number of subgroups G. The algorithm works in the same fashion as the first one. It starts by selecting the smallest neighbor set as a subgroup, then assigns the local distributor and deletes unnecessary information as described in the first algorithm. The procedure continues until no neighbor set is left. In the case of several neighbor sets having the same size and the same random number, regeneration and retransmission of random numbers are required to solve the conflict in deciding subgroups. Note that the results of running a grouping algorithm twice on the N users in a communication group may be different because of the random selection of subgroups among the largest neighbor sets.

Each user can use either of the grouping algorithms to determine the connection pattern in a distributed manner. For the sake of clarity, the term "users" refers only to the ordinary users and excludes local distributors. We evaluate the performance of the two grouping algorithms in Sect. 6.

2.3 The renegotiation process

We derive the hierarchical distribution architecture by using Eq. 8 and the grouping algorithms given in Sect. 2.2. Since the algorithm is distributed, each user is aware of the structure of the hierarchical architecture. To transform the direct connection to the hierarchical distributed architecture, each user starts the renegotiation process, which consists of two parts: (1) breaking the unnecessary connections to create the hierarchical topology and (2) renegotiating the bandwidth for some of the remaining connections to support the hierarchical architecture. Since all users know the hierarchical distribution architecture, they can determine which connections should be disconnected and which should be renegotiated. Breaking unnecessary connections is straightforward and can be done without difficulty. The bandwidth renegotiation for the remaining connections is explained in what follows.

The hierarchical distribution architecture consists of two types of connections (Fig. 5):

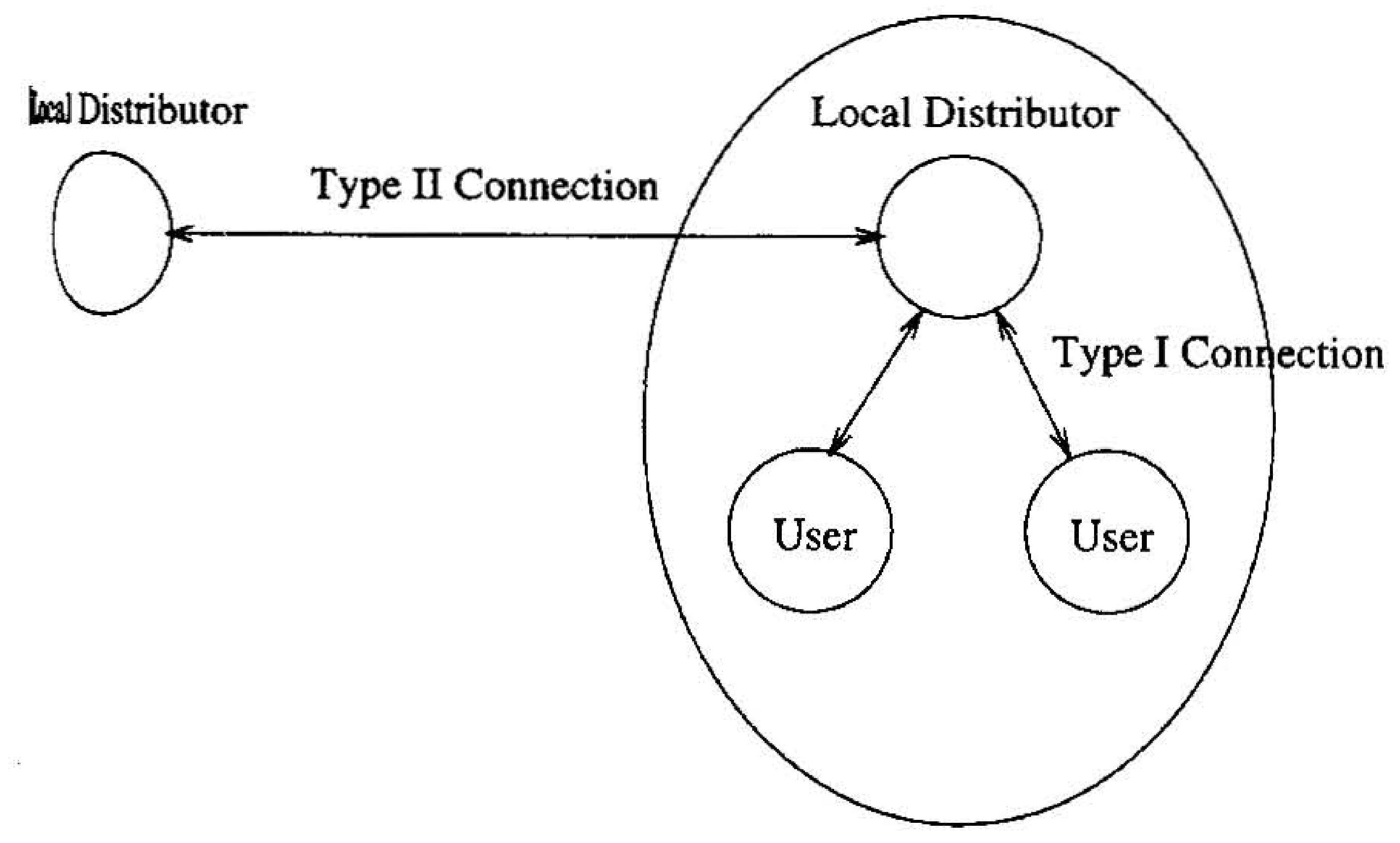


Fig. 5. Connection types

- *Type-I Connection.* Connections between a user and its local distributor: the original bandwidth B is large enough to carry media units from a user to its local distributor. However, at most $N \cdot B$ bandwidth is required to deliver the media units from the local distributors to the user.
- *Type-II Connection.* Connections between local distributors: if n_i is the number of users in the subgroup of a local distributor i, then the bandwidth required to carry media units from i to another local distributor, say j, is n_iB. The local distributor i receives media units from n_i users and transmits them to j.

Note that, in general, the bandwidths for a type-II connection is larger than the originally assigned bandwidth B. Moreover, for type-II connections, the maximum end-to-end delay jitter should be the same as originally negotiated because it was used to determine neighbor sets by users.

3 Computation of the buffer sizes

In general, the available buffer size γ_j at each user is not fully used. We need to compute the buffer size required so that we can release surplus buffer. The buffer size required to run a multimedia application at each user i is based on several factors, including the synchronization requirements of the application, the delay jitter, and the variation of propagation delays over various connections. In a telephony type of application, for example, the buffer is mainly used to compensate asynchrony caused by delay jitter. In this case, users care more for continuity of media units than for propagation delay, as long as it is within a few hundred milliseconds.

However, in time-critical applications such as teleorchestra, the buffer is used to overcome the asynchrony caused by the delay jitter and the variation of propagation delays over various connections. The synchronization requirement of time-critical applications demands that the media units produced by the users at exactly the same time or almost the same time should be displayed at all users at the same time. In a teleconference, several users must use the network to transmit the video and voice media units in real time. These media units from users are temporally related and must be played in an appropriate order, i.e., media units collected at the same time, or almost the same time, must be played back at the same time. To fulfil this requirement, media units that experience shorter end-to-end delay are buffered and must wait for the ones that experience a longer end-to-end delay.

In conclusion, the buffer size required at each user in an application of the telephony type is smaller than that required in time-critical applications. Thus, the buffer size in time-critical applications is the worst case for any type of application and is derived in Sects. 3.1 and 3.2. For the sake of clarity, we assume that the buffers are located at receiving sites.

3.1 Buffer sizes for users in each subgroup

Suppose a local distributor i sends multimedia traffic to its group member j through several connections with a bandwidth B and a maximum end-to-end delay jitter $J_{i,j}(m)$ of the connection carrying medium m. It is well known that one way to compensate for the effect of delay jitter is to prefetch buffering with the size of $\beta_{i,j}(m)$ given by the following formula (Ramanathan and Rangan 1993).

$$\beta_{i,j}(m) = \left\lceil \frac{J_{i,j}(m)}{\mu(m)} \right\rceil \tag{9}$$

where $\mu(m)$ is the playback period of the medium m. Thus, user j will not start playback of the media units from the local distributor i until all prefetched buffers $\beta_{i,j}(m)$ are full. Thus, the prefetch time $L_{i,j}$ measures the interval from the time when local distributor i starts to collect media units to the time when user j can start playback.

$$L_{i,j} = \max_m \{\Delta_{i,j}(m) + \beta_{i,j}(m)\mu(m)\} \ . \tag{10}$$

User i must still wait until all users in the subgroup complete prefetched buffering because of the synchronization requirement imposed by time-critical applications. Thus, the initial playback time M_i of the subgroup at which all the users in the subgroup start playback can be determined by the following formula:

$$M_i = \max_j \{L_{i,j}\} \tag{11}$$

The users in the subgroup keep buffering the media units from the local distributor i until the subgroup initial playback time M_i is reached, then start to play the media units back. The buffer size $\beta_{i,j}$ at each user j in the same subgroup with the local distributor i can then be calculated as follows:

$$\beta_{i,j} = \sum_m (M_i - \delta_{i,j}(m))Nb_m \tag{12}$$

3.2 Buffer sizes for local distributors

Suppose there is an algorithm that synchronizes the global initial collection time G_c and the global initial playback time G_p of the users in one communication group. (The global initial playback time G_p is not necessary for some applications such as telelecture. Again, the assumption allows us to find out the worst-case buffer-size requirement.) In Sect. 4, we give an example of such an algorithm. After the global initial collection time G_c, a local distributor i receives and buffers the temporally related media units from the users in the same subgroup before sending them to other local distributors. Furthermore, a local distributor also receives and buffers the temporally related media units from other local distributors and sends them to the users in the same subgroup. For later playback at i, the temporally related media units stay in the buffer of i after transmission to other users/local distributors. At time G_p, all users and local distributors start playing back. Moreover, local distributor i must send media units from its own subgroup in a synchronous fashion. However, temporally related media units from users in the same subgroup experience different delays over the connections to their local distributor i. To send the media units synchronously, the prefetched buffering time $\overline{M_i}$ is needed for i to buffer the media units from all users in the same subgroup. $\overline{M_i}$ is computed by

$$\overline{M_i} = \max_j \{L_{j,i}\} \tag{13}$$

In other words, at the time $G_c + \overline{M_i}$, the local distributor i starts to transmit media units of the subgroup to other local distributors at steady rates. Thus, the buffer size required at a local distributor i can be obtained from:

$$\beta_i = \sum_j \sum_m \{(\overline{M_i} - \delta_{j,i}(m))b_m\} + \sum_k \sum_m \{(G_p - G_c - \overline{M_k} - \delta_{k,i}(m))n_k b_m\} \tag{14}$$

where n_k is the number of users in the subgroup k including the local distributor in the group and b_m is the average bandwidth required to transmit on the medium m. The first term on the right-hand side of Eq. 14 refers to the buffer size required for the local distributor i to store the media units from the users in the same subgroup. The second term refers to the buffer size needed for i to store the media units from other local distributors of the application. In Sect. 6, we compare the buffer sizes in the hierarchical distribution architecture and the direct connection.

The total buffer size β_h required is given by

$$\beta_h = \sum_{i=1}^{G} (\beta_i + \sum_{k=1}^{n_i - 1} \beta_{i,k}) \tag{15}$$

Note that the subscript h of β_h indicates that the hierarchical distribution algorithm is used. The total buffer size β_d in direct connection can also be calculated by using Eqs. 14 and 15. In the computation of β_d, let $G = N$ and consider each user as a local distributor.

4 Synchronization issues

In time-critical multimedia applications, temporally related media units are played by all users at the same time. To fulfill the synchronization requirement, we must solve four problems. They are *local clock drift, delay jitter effect, global initial collection time* G_c and *global initial playback time* G_p. Let us assume that there is a global clock in the network. We can solve the delay-jitter problem by appropriate prefetched buffering (Ramanathan and Rangan 1993). In this section, we demonstrate the derivation of G_c and G_p (Akyildiz and Yen 1995; Yen and Akyildiz 1994).

G_c and G_p can be determined by the local distributors in a distributed manner. In the renegotiation processes, each local distributor i sends M_i to other distributors of the application. At the end of the renegotiation processes, the local distributor i sends the time I_i to other distributors as a candidate for G_c. Time I_i has a lower bound that assures that all users can receive it on time.

$$I_i \geq \tau_i + \max_{m,j}\{\Delta_{i,j}(m) + M_j\} \tag{16}$$

where τ_i is the sending time of I_i, and j refers to all local distributors. If I_i satisfies Eq. 16, all users in the communication can get I_i on time, and G_c is determined by

$$G_c = \max_i\{I_i\} \tag{17}$$

To obtain G_p we first derive λ, the *preparation time*, required for all users to fill their prefetched buffers after G_c, so that G_p becomes equal to $G_c + \lambda$. The preparation time λ is determined by:

$$\lambda = \max_{i,j}\{\overline{M_i} + L_{i,j} + M_j\} \tag{18}$$

The interpretation of Eq. 18 is straightforward. For a local distributor i, time $\overline{M_i}$ is required for i to buffer the media units from the users in the same subgroup after G_p. After the prefetched buffering time $\overline{M_i}$, the local distributor i starts to send media units to another local distributor j and it takes time $L_{i,j}$ for j to prefetch the media units from i before multicasting them to the users in the same subgroup with j. Finally, the local distributor j starts to send media units to the users in the same subgroup. It takes time M_j for the users in the subgroup to become ready to play back. Since every user is required to start playing at G_p, the preparation time λ is obtained by taking the maximum value with respect to every pair of local distributors (i, j). Note that λ is available before the users of the application reach consensus on G_c, so that the local distributors can use λ to estimate the buffer sizes.

5 Architecture management

Now we have presented the complete algorithm that transforms the communication architecture from the direct connection to the hierarchical distribution architecture. We have also addressed the synchronization issues of the hierarchical architecture. However, if we do not handle the entering and leaving of users and local distributors carefully, it can affect the stability of the architecture. We introduce a reconfiguration algorithm that solves the problem of users entering and leaving the architecture, resulting in the least effect on the existing communication.

A new user x can join the communication architecture by first connecting to an existing user i. Since the grouping algorithm is distributed, each user has full knowledge of the grouping information such as the list of local distributors. Thus, user i can give the addresses of the local distributors to the new user x, so that x can determine group membership. The new user x, after getting the addresses of the local distributors, makes connections with all local distributors and negotiates the maximum end-to-end delays with each of them separately. The prefetch time $L_{k,x}$ of the connection from any of the local distributor k to x can be calculated once

Table 1. Comparison of total buffer sizes in direct and hierarchical connections

Number of users N	Available buffer (only for hierarchical case) γ_j Mbits	Average total buffer size (direct) $\overline{\beta_d}$ Mbits	Average total buffer size (hierarchical) $\overline{\beta_h}$ Mbits
5	1	13.47	11.46
10	2	67.10	44.27
20	4	292.74	139.68
50	10	2118.5	627.1

the maximum end-to-end delays $\Delta_{k,x}(m)$ are negotiated. As mentioned in Sect. 2.1, if $L_{k,x}$ is less than the threshold α_k of the prefetch time, then the new user x is qualified to join the subgroup associated with the local distributor k. However, if there is more than one subgroup for which x can join, the new user x randomly selects one subgroup and breaks the connections with other local distributors. In the extreme case, if x is not qualified for any subgroup, then x becomes a local distributor and communicates with other local distributors directly.

To leave the architecture, a user informs the associated local distributor and breaks the connections. However, if a local distributor i wants to leave the communication, then two procedures must be carried out. (The failure at the local distribution can be treated as a special case of leaving.) First, the remaining users of the subgroup must be divided into smaller groups depending on their locations. Users of each subgroup must elect one local distributor and establish connections to it. Also, the newly elected local distributors must establish connections to other local distributors. Once the users know that the local distributor i in the same subgroup is leaving, the election of new local distributors is initiated, and each of them starts to make connections with other users in the subgroup. The grouping algorithms introduced in Sect. 2 are run in the subgroup, so that new subgroups and associated local distributors are generated. It is possible that more than one subgroup can be generated. Without loss of generality, assume that two new subgroups and associated local distributors k and l are selected by the grouping algorithms. A user j in the same subgroup with the local distributor i belongs to the subgroup associated with either the local distributor k or the local distributor l. For the sake of clarity, we assume that j belongs to the subgroup associated with the local distributor k. Once the new local distributors k and l are elected, they try to make connections with each other and the existing local distributors. Note that each user must update the grouping information including the list of local distributors and their group members, every time the architecture changes.

After the connections are established, user j can get media units through the old local distributor i or through the new local distributor k. Thus, the local distributors k and l can start to prefetch media units with time stamp ν from the other local distributors and the users in respective subgroups. The time stamp ν is decided in the same way as the global initial collection time G_c is generated. The remaining problem is how long it will take for k to buffer enough media units so that the delay jitter effect of the connections associated with k can be compensated.

Table 2. Average number of subgroups and average buffer sizes of local distributors and users

Number of users N	Average number of subgroups G	Average buffer size of local distributors $\overline{\beta_i}$ Mbits	Average buffer size of users $\overline{\beta_{i,j}}$ Mbits
5	3.49	3.07	0.48
10	5.2	7.56	1.03
20	6.61	16.58	2.249
50	8.21	45.39	6.088

Table 3. Buffer efficiency for various buffer sizes γ_j in the hierarchical architecture

Available buffer size γ_j Mbits	Total buffer size $\overline{\beta_h}$ Mbits	Buffer size – local distributors $\overline{\beta_i}$ Mbits	Buffer size – users $\overline{\beta_{i,j}}$ Mbits	Number of subgroups G
2	268.23	15.06	1.45	17.58
4	139.80	16.64	2.27	6.57
8	128.30	17.42	4.58	2.86
16	201.8	30.50	9.02	1

As mentioned in Sect. 4, the difference between G_c and G_p is λ, i.e., if a media unit is sent at time ξ, then it will be played by all users at time $\xi + \lambda$. Since the value of λ depends on the set of local distributors in the application, it must be recalculated by using Eq. 18 whenever the set of local distributors is changed, i.e., when a new user joins the architecture as a local distributor or when an existing local distributor wants to leave. Note that the recomputed λ value is not necessarily different from the previous one. However, to distinguish between two λ values in different periods of the communication, we call the original λ value λ_{old} and the value after recalculation as λ_{new}. It is obvious now that at the time $\nu + \lambda_{new}$, the communication group is ready to release the local distributor i and the associated connections. The new local distributors k and l start to serve the users originally in the subgroup of i. There is no disruption of playback when a new user joins the communication group or when a user leaves the group, except when the local distributor i leaves the communication group before $\nu + \lambda_{new}$.

The communication groups can be set up in two ways. First, all users of the group start to build full connections with one another and apply the grouping algorithm to establish the hierarchical architecture. Second, some users of the group can initiate the communication and allow the remaining users to join the communication later by using the group management protocol already mentioned. Note that, given a set of users, the first and the second methods may generate different connection patterns among the users.

6 Performance evaluation

By using simulation (carried out on a single workstation) we investigate the performance of the hierarchical distribution architecture. Our simulation model consists of N users. The location of each user is represented by randomly generated (x, y) coordinates that are uniformly distributed on a square. Each side of the square is equal to 100 units. The distance between two users represents the minimum end-to-end delav

Table 4. Comparing the performance of the two grouping algorithms. The indices 1 and 2 are associated with the first and the second grouping algorithms, respectively. N, number of users; γ_j, available buffer size; G, number of subgroups; B_h^*, total bandwidth; H_h, number of connections; $\overline{\beta_h}$, average total buffer size; $\overline{\beta_i}$, average buffer size of local distributors

N	γ_j	$G(1)$	$G(2)$	$B_h^*(1)$	$B_h^*(2)$	$H_h(1)$	$H_h(2)$	$\overline{\beta_h}(1)$	$\overline{\beta_h}(2)$	$\overline{\beta_i}(1)$	$\overline{\beta_i}(2)$
5	1	3.59	3.77	44.19	43.82	12.12	12.9	11.84	12.12	3.14	3.85
10	2	5.22	6.37	195.63	193.25	31.59	41.47	43.86	51.52	7.43	7.56
20	4	6.52	8.92	812.14	807.19	62.95	93.07	139.94	190.12	16.77	18.85
50	10	8.21	12.88	5143	5133.42	142.77	227.25	631	911.01	45.85	54.54

of the connections between them. Each user wants to communicate with other users by sending and receiving two media streams, video and voice. The playback period for video is 33 ms and for voice, it is 10 ms. The bandwidth B for the virtual connections carrying the two media from one user to another requires approximately 2 Mbits/s. (Compressed video requires 2 Mbits/s and voice requires 64 kbits/s.) The maximum delay jitter of a connection is assigned randomly, and it ranges from 10% to 60% of the minimum end-to-end delay of the connection. The simulation results given in Tables 1–3 are obtained from the first grouping algorithm in Sect. 2.2. In Table 4, we compare the two grouping algorithms.

We use the first grouping algorithm to divide the users into several subgroups and calculate the buffer size for each user with Eqs. 12 and 14. For each N, we generated 100 sets of locations of users and calculated the total buffer size β_h from Eq. 15 accordingly. In Table 1 we compare the total buffer size β for the cases of direct and hierarchical connections.

In Table 1, we increase the available buffer size γ_j as the number of users N increases. This is because the temporally related media units increase linearly with N and must be stored to maintain the synchronization. Later we show the effect of different available buffer sizes, γ_j, by keeping the number of users N constant. In Table 1, we demonstate that our architecture saves more and more buffer size as the number of users N increases. This phenomenon can be explained by the number of subgroups G listed in Table 2. Note that the available buffer size γ_j is only for users and not for the local distributors. In general, the buffer size for the local distributor is substantially larger than the buffer size for each user. The total buffer size for the hierarchical architecture β_h is the of the sum of the buffer sizes of all users and all local distributors.

As shown in Table 2, the number of subgroups G increases much more slowly than the number of users
the size of each subgroup becomes larger when N in
In other words, more users can share the buffer of o
distributor; higher buffer efficiency can be achieve
increases.

In Table 2, we also show the average buffer size
distributors and ordinary users. Note that the averag
size of ordinary users $\overline{\beta_{i,j}}$ is roughly one-half the a
buffer size γ_j. This ratio results from the random
the locations of users with respect to the associate
distributor. In other words in one subgroup, some u
relatively closer to the local distributor than the ot
that the users close to the local distributor must sto
media units to wait for the user who is relatively far away.

In conclusion, although the average buffer size of ordinary users is one-half the available buffer size γ_j, some users must use all of the buffer γ_j while others may only use a small portion of the buffer. However, after the communication is set up, the users can release the spare buffer size $(\gamma_j - \beta_{i,j})$.

The average buffer size of local distributors is roughly equal to the average buffer size of the users in the direct connection case, which is given by $\frac{\beta_d}{N}$. Thus, our architecture significantly reduces the buffer size of ordinary users without increasing the buffer size of local distributors, in comparison with the case of direct connections.

In Table 3 we vary the available buffer size γ_j while keeping $N = 20$, and show the performance. As also shown in Table 3, increasing the available buffer size γ_j does not necessarily improve the buffer efficiency. As γ_j increases to 16 Mbits, all users are in one subgroup. According to Eq. 8, the larger γ_j is, the larger α_i is. When γ_j becomes larger, all neighbor sets S_i are the same and contain all users. According to the first grouping algorithm, the local distributor is determined by the random number r_i, i.e., the user i with the largest r_i becomes the local distributor. Thus, the selection of local distributor is probably not optimal, and this causes a large buffer size at the local distributor to synchronize media units from all users. Since all neighbor sets contain all users, no matter who the local distributor is, the available buffer size γ_j is sufficient to buffer media units.

In Table 4 the first grouping algorithm minimizes the number of groups so that it has fewer number connections H_h and results in a smaller buffer size $\overline{\beta_h}$, while the second one achieves less total bandwidth B_h^*.

7 Conclusion

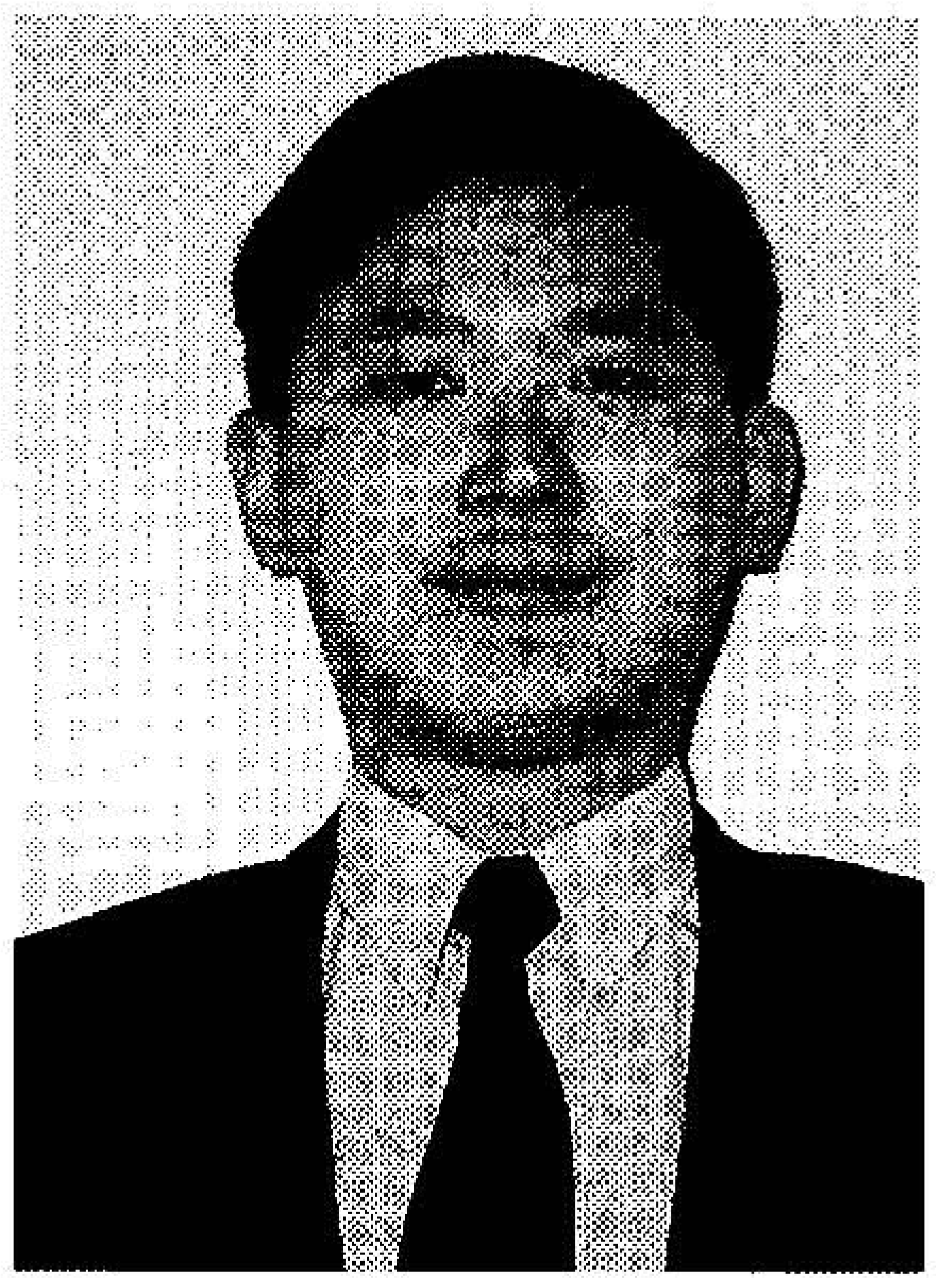
In this paper, we developed a hierarchical architecture for multimedia distribution. Although the analysis and experiments are based on n-n configuration applications such as teleconference, we can apply the architecture to a 1-n configuration with a slight modification. The hierarchical architecture is established with a distributed grouping algorithm. In our model, we do not assume, that the multicast function is implemented in the nodes of the networks. However, a good point-to-point routing algorithm is required.

With the hierarchical architecture, we reduce the total buffer size β_h significantly without suffering serious bandwidth and delay penalty. The buffer sizes of local distributors in the hierarchical architecture are about the same as those of the users in direct connection architecture. In other words, the hierarchical architecture does not introduce additional hot spots in terms of buffer size. Furthermore, if media are mixed at the local distributors, then we can save more buffer space and bandwidth.

Since the number of users may change within the lifetime of the communication group, we introduce an architecture management that allows the users to join or leave the communication group with the least effect on the existing communication. In Core Based Tree (CBT) or other minimum Steiner tree-based multicast routing algorithms, it seems that these multicast routing algorithms must be used every time the membership of the communication group changes. In other words, these algorithms require the involvement of all users whenever a user joins or leaves the group. In contrast, in the worst case, our management functions require only the use of the grouping algorithm within a subgroup and the involvement of all local distributors.

References

1. Adam JF, Houh HH, Ismert M, Tennenhouse DL (1994) A network architecture for distributed multimedia systems. Proceedings of ICMCS'94, Boston, May 1994
2. Akyildiz IF, Yen W (1995) Multimedia group synchronization protocols for integrated networks IEEE J Selected Areas Commun, Vol. 14, No. 1, January 1996
3. Bolot JC (1993) End-to-end packet delay and loss behavior in the internet. Proceedings of the ACM SIGCOM '93 Conference, San Francisco, pp 289–298, Sept. 1993
4. Escobar J, Partridge C, Deutsch D (1994) Flow synchronization protocol IEEE/ACM Trans Networking, April 1994, pp 111–121
5. Ramanathan S, Rangan PV (1993) Adaptive feedback techniques for synchronized multimedia retrieval over integrated networks. IEEE/ACM Trans Networking 1:246–260
6. Rangan PV, Vin HM, Ramanathan S (1993) Communication architectures and algorithms for media mixing in multimedia conferences. IEEE/ACM Trans Networking 1:20–30
7. Yen W, Akyildiz IF (1994) On the synchronization mechanisms for multimedia integrated services networks. Proceeding of the COST 237 Conference on Multimedia Transport and Teleservices, Springer, Berlin Heidelberg New York, November 1994



WEI YEN was born in Kaohsiung, Taiwan. He received his BS from the Tatung Institute of Technology, Taiwan, in 1989, and his MS from the Georgia Institute of Technology in 1993. He is currently a PhD student in the Department of Electrical and Computer Engineering at the Georgia Institute of Technology. His current research interests include communication protocol design and analysis for multimedia applications, multicast routing, and ATM networks. He is a student member of IEEE.

IAN F. AKYILDIZ received his BS, MS, and PhD degrees in Computer Engineering from the University of Erlangen-Nürnberg, Germany, in 1978, 1981, and 1984 respectively. Currently, he is a Professor with the School of Electrical and Computer Engineering, Georgia Institute of Technology. Dr. Akyildiz is a senior member of the IEEE, a member of ACM (SIGCOMM) and a National Lecturer for ACM. He is an editor for IEEE Tr. on Computers, ACM-Baltzer Journal for Wireless Networks and Computer Networks and ISDN Systems Journal.