Proceedings of the 17th IASTED International Conference
Parallel and Distributed Computing and Systems
November 14-16, 2005, Phoenix, AZ, USA

HIERARCHICAL CLUSTERING OF MESSAGE FLOWS IN A MULTICAST DATA DISSEMINATION SYSTEM

Yoav Tock, Nir Naaman, Avi Harpaz, Gidon Gershinsky
IBM Haifa Research Laboratory
Mount Carmel, Haifa 31905, Israel
{tock,naaman,harpaz,gidon}@il.ibm.com

ABSTRACT
A large-scale data dissemination application is characterized by a large number of information flows and information consumers. Consumers are interested in different, yet overlapping, subsets of the flows. Multicast is used to deliver subsets of the flows to subsets of the consumers. Since multicast groups are a limited resource, each consumer must filter out a large number of unneeded flows. We alleviate the end-node filtering load by using hierarchical clustering of flows to transport-layer sessions, and clustering of sessions to network-layer multicast groups. This scheme allows for hierarchical filtering of flows at the receivers. We formulate a cost function that models and emphasizes the filtering process, and propose algorithms for the solution of the hierarchical mapping problem. Performance evaluation indicates a significant reduction of end-node filtering cost compared to a non-hierarchic approach.

KEY WORDS
Multicasting algorithms, multicast mapping, data dissemination, receiver interest, hierarchical clustering, optimization algorithms.

1 Introduction

Consider a large-scale data dissemination application that is characterized by a large number of information flows (in the hundreds of thousands), and a large number of information consumers (in the thousands). Each information flow generates messages which must be delivered to interested consumers. Information consumers display interest heterogeneity, that is, consumers are interested in different, yet overlapping, subsets of the information flows. Naturally, an individual information flow may be required by many consumers.

Such a setup is typical of a large financial trading office, for example, where the flows can be stock quotes, commodity prices, etc., and the consumers can be traders, analysts and so on. Each trader or analyst is interested in a different portfolio — thus displaying interest heterogeneity across the data consumers. A simplified model of such a system is shown in Fig. 1. The publisher divides the data feed into a large number of topics (a synonym to information flows), and each consumer subscribes to his topics of interest. We assume that the publish-subscribe part of the system is confined to a multicast-enabled enterprize LAN.

The challenge is to deliver the messages generated by the flows to the interested consumers in an efficient manner. In a sparse yet correlated subscription pattern [1], such as the one we assume, flooding is very inefficient as the consumers will be burdened heavily with an enormous amount of unwanted incoming traffic. Unicast, on the other hand, is perfect for the consumer, but many messages will travel multiple times on the common parts of the network, wasting network resources and heavily loading the transmitter.

It was suggested by [2] to solve the message distribution problem by assigning a multicast group per flow. However, multicast groups are a limited network resource, because routers must save and maintain state information for every multicast group used. Moreover, certain end-node systems pose limitations on the number of multicast groups one can join. Thus, using a multicast group per flow is impractical for large scale systems. An alternative is to map the large number of flows to a fixed number of multicast groups, and to assign each receiver with a set of multicast groups so as to satisfy its flow subscriptions. The problem is to find these pair of mappings so as to minimize some cost function that quantifies system performance. This had been termed the "*channelization*" problem and shown to be NP-hard by [3]. Several authors have tried to tackle this problem by clustering flows into multicast groups [4, 5, 1].

A solution to the channelization problem, according to the cost function proposed by [3], aims to strike a balance between the total bandwidth consumed and the amount of unwanted information received by consumers. Thus, in general, even the optimal solution still leaves the consumers with the need to filter the incoming stream of messages. It has been found [6] that in a high throughput messaging application over a fast enterprize network, it is often the computing power of end-nodes that limits performance. The fact that the number of flows is orders of magnitude larger than the practical number of multicast groups ($\sim 10^5$ vs. $\sim 10^2$), together with the large number of receivers ($\sim 10^3$), aggravates this problem.

Our main goal is to further reduce the filtering load imposed upon the receivers. To that end, we introduce a hierarchical clustering scheme that allows for hierarchical filtering of flows at the receivers. We propose to cluster the flows into transport-layer multicast sessions, and cluster the

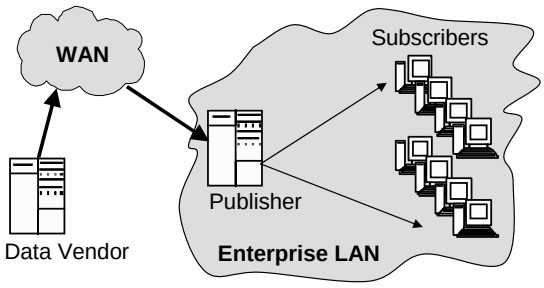

Figure 1. A simplified model of a financial data dissemination system.

sessions into network-layer multicast groups. We formulate a cost function that models and emphasizes the hierarchical filtering process, and incorporate it in algorithms for the solution of the hierarchical mapping problem. A statistical model for consumer interest and message rate, based on real-life data from the financial domain, is presented. The statistical model is used to evaluate the performance of the proposed scheme.

Finally, let us remark that alleviating the filtering burden off the receivers has also been the goal of content-based messaging. However, central filtering had been deemed slow, and broker-assist solutions (e.g. [7], [2]) introduce delays in the data path, which makes them inapplicable in certain scenarios. On the other hand, multicast is now widely available and is enabled by default in most LANs. We thus see an advantage in utilizing multicast capabilities for high throughput messaging.

2 Problem Description and Model

Let F denote the set of information flows, $|F| = K$. Flow F_k produces a sequence of messages with rate λ_k messages per second, $k \in F$, and $\lambda = [\lambda_1, \cdots, \lambda_K]$.

Let U denote a set of users (consumers), $|U| = N$. Each user U_n contributes a binary "interest vector" of length K, where a 1' in the k^{th} position denotes his interest in flow F_k. The rows of the "interest matrix" $W = (w_{nk})$, $k \in F$, $n \in U$, are the users' interest vectors:

$$w_{nk} = \begin{cases} 1 & \text{user } U_n \text{ is interested in flow } F_k \\ 0 & \text{otherwise} \end{cases}$$

Each flow is mapped into a session (also referred to as a "stream"), which is a globally unique transport layer entity, for example, a transport session in Pragmatic General Multicast (PGM) [8, 9]. Each session is mapped to a multicast group (see Fig. 2). Let S denote the set of sessions (streams), $|S| = L$; and G denote the set of multicast groups, $|G| = M$.

In the general case, a session might be mapped to more than one multicast group. However, in order to avoid the complications associated with stream duplication, in this work we restrict each session to be mapped to a single multicast group. This restriction is in accordance with the specifications of most reliable multicast protocols (e.g., PGM).

The flow-to-session mapping matrix, $X = (x_{kl})$, $k \in F$, $l \in S$, is defined

$$x_{kl} = \begin{cases} 1 & \text{flow } F_k \text{ is mapped to session } S_l \\ 0 & \text{otherwise} \end{cases}$$

Let the total rate of session S_l be θ_l messages per second, and $\theta = [\theta_1, \cdots, \theta_L]$. That is, $\theta = \lambda \cdot X$. The session-to-group mapping matrix, $Y = (Y_{lm})$, $l \in S$, $m \in G$, is defined

$$y_{lm} = \begin{cases} 1 & \text{session } S_l \text{ mapped to group } G_m \\ 0 & \text{otherwise} \end{cases}$$

A user interested in an information flow must listen to the appropriate multicast group, pull out the relevant session, and extract the desired information flow from the session. See for example Fig. 2, where user U_3, interested in flow F_3, might be given the reverse path $/G_m/S_l/F_3$ (note that there is more than one reverse path from U_3 to F_3).

The subscription matrix, $Z = (z_{nl})$, $n \in U$, $l \in S$, specifies to which sessions must each user subscribe

$$z_{nl} = \begin{cases} 1 & \text{user } U_n \text{ subscribes to session } S_l \\ 0 & \text{otherwise} \end{cases}$$

The group listening matrix, $P = (p_{nm})$, $n \in U, m \in G$, specifies to which multicast groups must each user join

$$p_{nm} = \begin{cases} 1 & \text{user } U_n \text{ joins group } G_m \\ 0 & \text{otherwise} \end{cases}$$

Since each session is mapped to a single multicast group, $P = u(Z \cdot Y)$, where $B = u(A)$ is the point-wise step operator, $b_{i,j} = 1$ for $a_{i,j} > 0$, and 0 otherwise.

A *legal* set of mappings must comply to several requirements. Specifically:
(i) *No false exclusion* — all the flows a user is interested in are mapped to one or more sessions to which the user subscribes. That is,

$$\sum_{l \in S} z_{nl} \cdot x_{kl} - w_{nk} \geq 0. \quad \forall \quad n \in U,\ k \in F.$$

(ii) *No dummy sessions or groups* — no empty, unmapped, or un-listened sessions and groups. That is,

$$\sum_{k \in F} x_{kl} > 0 \quad \forall\, l \in S, \quad \text{and} \quad \sum_{l \in S} y_{lm} > 0 \quad \forall\, m \in G.$$

This is a parsimony requirement.

The *optimal* set of mappings must also minimize a certain cost function. Thus, the problem can be phrased in the following way. Given F, λ, S, G, U, W, find a set of mappings X, Y, Z, that complies with the constraints, and minimizes the cost function $C(X, Y, Z)$. In this paper

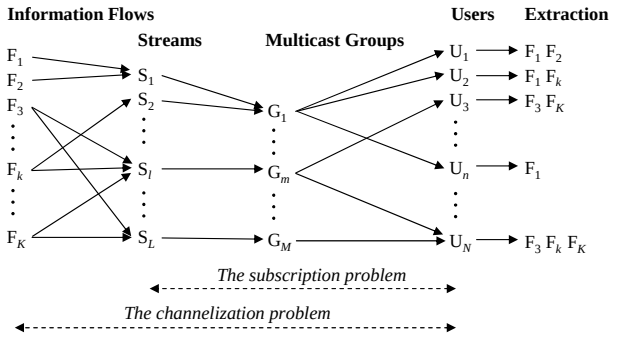

Figure 2. Mapping between information flows, streams, multicast groups, and users (with flow duplication).

we propose a cost function that models the filtering cost incurred upon the users (defined in the sequel).

Let us call the overall three-stage mapping problem from information flows to users the "channelization problem", and the user-to-session mapping the "subscription problem" (as in [3], see Fig. 2). Note that in our approach, the mapping from information flows to multicast groups is a two stage process, whereas in [3] it is a single stage process, in which sessions do not exist, or are synonymous with flows.

The flow-to-session mapping can be constrained to exclude a mapping that maps a flow to more than one session, that is, $\sum_{l \in S} x_{kl} \leq 1$. This is called the "*no-duplication*" constraint. It has been shown [3] that the original channelization problem is NP-hard in both the constrained and unconstrained cases; whereas the subscription problem, which is NP-hard in the unconstrained version, can be solved in linear time in the constrained case. It can be shown that, despite the differences in structure and cost function, the above mentioned results also apply to our version of the channelization and subscription problems.

3 Mapping Costs

The mapping costs can be classified into two categories: network resources, and processing resources. Network costs such as total transmission rate, end node weighted reception rate, and end node excess rate were treated by [3]; average receiver goodput was treated by [4, 5]. The said cost functions *implicitly* take into account the processing load incurred upon the receivers. As discussed earlier, our goal is to alleviate the end-node processing load. We therefor pay a closer attention to the processing costs in the hierarchical filtering scheme we proposed.

3.1 Processing Costs

The filtering process is made of two stages. Superfluous sessions that belong to a multicast group the user is listening to are dropped first. Superfluous flows that belong to a session the user is subscribed to are dropped second (see Fig. 3). Note that sessions are filtered by a lower layer than the flows (transport layer versus messaging layer), and that it is always better to filter an excess message at the lowest possible layer. Moreover, messages sent on multicast groups the user is *not* listening to are filtered by the network elements and network interface card, with no cost to the user. The transition to filtering in two stages can increase the filtering speed because filtering streams is faster than filtering flows. In addition, a substantial reduction in filtering costs comes from another technique we call *message aggregation*.

3.2 Message Aggregation

In many messaging applications, and in financial data dissemination systems in particular, messages are usually quite short. Message lengths of 100B to 1KB are typical. Thus, it is possible to aggregate several messages into a single network packet. This technique had been shown to produce significant savings in packet processing overhead, since some of the fixed cost of processing a packet is amortized over all the messages in that packet [6]. Let us assume each packet belongs to a single session. Denote by h the "aggregation factor" — the average number of messages per packet (see Fig. 3). Let us quantify the processing load per message.

Session filtering cost — For every session a user receives, a cost $\alpha(h) \cdot \theta_l$ is added.

$$C_1^h = \sum_{n \in U} \sum_{l \in S} \sum_{m \in G} p_{nm} \cdot y_{lm} \cdot \alpha(h) \cdot \theta_l \quad (1)$$

$$\alpha(h) = \alpha_0 + \alpha_1 / h \quad (2)$$

Flow filtering cost — For every flow in a session that is actually being accepted by the user, a cost $\beta(h) \cdot \lambda_k$ is added. Thus,

$$C_2^h = \sum_{n \in U} \sum_{l \in S} \sum_{k \in F} z_{nl} \cdot x_{kl} \cdot \beta(h) \cdot \lambda_k \quad (3)$$

$$\beta(h) = \beta_0 + \beta_1 / h \quad (4)$$

Given that the messages are of fixed length, α_0 and β_0 are elements that depend on the packet length, whereas α_1 and β_1 are elements associated with a fixed cost per packet.

Note that subject to the no-false-exclusion requirement, minimizing $C_w = C_1^h + C_2^h$ amounts to minimizing the extra filtering work due to the superfluous streams and flows received by the users, denoted C_x.

4 Clustering Algorithms

In the general case, one can map a flow to more than one stream. We limit our attention to the "*no-duplication*" case, where a flow can be mapped to a single session. This makes the subscription problem solvable in linear time [3]. Since messages are not duplicated, transmission rate is kept

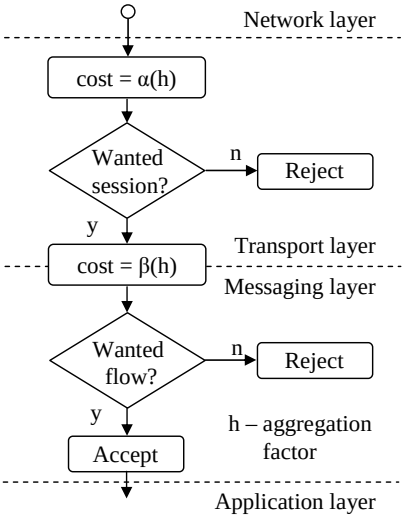

Figure 3. Hierarchic filtering costs.

to a minimum, saving transmitter and network resources. Moreover, the no-duplication constraint significantly reduces the management costs of the system. This is especially important for a large scale system like the one we explore.

The "no-duplication" constraint allows mapping flows to streams to be viewed as *clustering* flows to streams, and the same apply for mapping steams to groups. A simple heuristic for building the clustering hierarchy is to take the results of clustering in one level as the input of the next. Several variants of this approach will be presented in the sequel.

Since the cost functions (1) and (3) of the two hierarchies are identical in structure, we describe the clustering routine for one level only. We now describe a distance measure derived from C_1^h and C_2^h that will form the heart of the clustering algorithms that follow.

4.1 A "Distance" Measure

Every flow, F_k, is associated with a feature vector V_k and a message rate λ_k. The binary vector V_k is defined to be the k^{th} column of the interest matrix W. The coordinates of V_k are "users" — thus, flows can be considered to be points in "user-interest space". The "distance" between two points (flows) is defined as

$$d^p(F_i, F_j) = d^p(V_i, V_j) = \sum_{n \in U} g(V_j(n) - V_i(n), \lambda_i, \lambda_j) \tag{5}$$

where

$$g(x, \lambda_1, \lambda_2) = \begin{cases} 0 & x = 0 \\ \lambda_1 & x > 0 \\ \lambda_2 & x < 0 \end{cases} \tag{6}$$

Function $d^p(F_i, F_j)$ quantifies the amount of excess filtering incurred upon the users if F_i and F_j are clustered into the same stream. (Note that (5) is not a proper distance measure as it does not maintain the triangle inequality.)

Recall that stream S_l is a set of flows, $S_l = \{F_i, F_j, \cdots\}$, the total rate of which is θ_l. The *centroid* of the stream is defined as

$$C_l\{S_l\} = \bigoplus_{F_k \in S_l} V_k, \tag{7}$$

where $\bigoplus$ denotes the bit-wise OR of binary vectors V_k. In other words, $C_l(n) = 1$ if user n subscribes to stream l, and 0 otherwise. The distance between flow F_k and stream S_l is defined as

$$d^g(F_k, S_l) = \sum_{n \in U} g\left(C_l(n)\{S_l - F_k\} - V_k(n), \lambda_k, \theta_l\right) \tag{8}$$

where $C_l(n)\{S_l - F_k\}$ means that we remove F_k from the stream it belongs to in order to calculate the distance to that stream, and $g(\cdot,\cdot,\cdot)$ is defined in (6). This function quantifies the amount of excess filtering incurred upon the users if F_k is added to stream S_l. Note that (8) is the "local" equivalent of (3), given the no-false-exclusion requirement.

4.2 K-Means

We use the K-means clustering algorithm [10], with flows and centroids (7) as points in "user-space", and (8) as a distance measure between them:

1) *Initialize*: associate each point to a group according to some policy (e.g. random), calculate the centroid (7) of each group.

2) *Nearest neighbor*: pick a point and reassign it to the closest group, using (8). Do not reassign in case of a tie.

3) *Centroid*: update the the centroids of the old and new groups for that point.

4) *Stop*: if one pass over all the points does not produce a group change then stop; else goto step 2.

Note that each step of the algorithm can only reduce the cost of the partition, as calculated by (3). Thus, convergence is guaranteed. However, the algorithm does not guaranty convergence to a global minimum [10]. The standard approach is to restart the algorithm several times (with random initialization) and choose the best outcome. The stop condition can be augmented by limiting the number of iterations, running time, or the improvement rate of the total cost.

4.3 Hierarchical Clustering Algorithms

Our goal is to cluster flows to streams and streams to groups. This can be achieved by using a one-level clustering algorithm in two stages. Several variants exist.

Streams First (SF): Cluster flows to streams, and then cluster the resulting streams into groups.

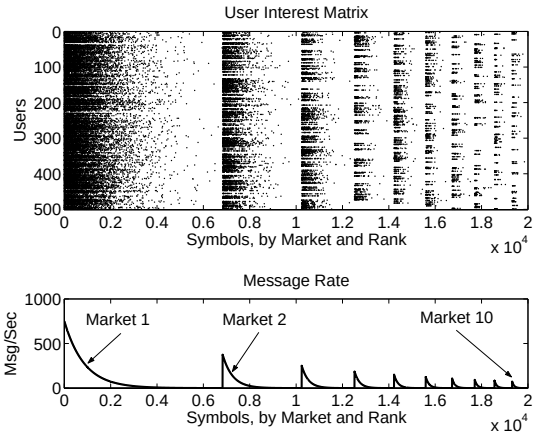

Figure 4. A realization of the receiver interest matrix, W, and message rate, λ.

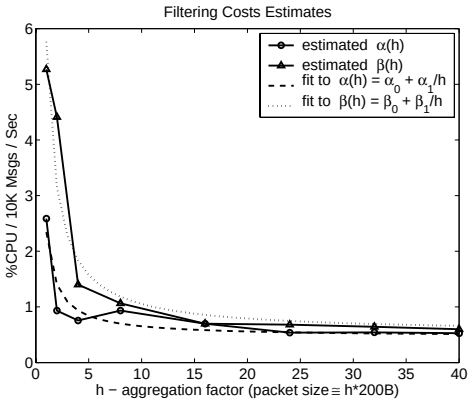

Figure 5. Estimation of processing costs as a function of the aggregation factor.

Groups First (GF): Cluster flows into groups, and then treat each group separately and cluster flows into streams within each group.

An Iterative Approach (IT): Iterative invocation of the GF and SF: 1) Start with a iteration of GF. 2) Use the resulting X and Y maps as a starting point for an iteration of SF. 3) Flatten the resulting two level map to a flow-to-group map and use it as a starting point for an iteration of GF. 4) Compute the cost of each new map (X,Y) and save it if it is better than the previous best map. 5) Continue to iterate from step 2, or stop after a prescribed number of iterations.

Random Restart with Annealing (RRA): 1) Fix $P_f = P_1$, $0 < P_0 < P_1 \leq 100$ and $0 < \mu < 1$. Start with an iteration of GF. 2) Choose randomly P_f percent of the flows, and reassign them randomly to the streams. 3) Use the randomized maps (X,Y) as a starting point for another round of GF. 4) Compute the cost of the map (X,Y) and save it if it is better that the previous best map. 5) Reassign $P_f = P_f \cdot \mu$, and repeat from step 2, or stop if $P_f < P_0$.

5 Performance Evaluation

5.1 Receiver Interest and Message Rate

In order to better understand the real environment in which a data dissemination application would be required to function, we studied the number of trades executed on each symbol (stock) during one day, over a period of one month, in the New York Stock Exchange (NYSE) [11]. We assume this number is proportional to the interest of users in the respective stock, as well as the message rate associated with each symbol. The empirical daily trade distribution was fitted quite accurately with an exponential curve, with minor variations from day to day. The normalized cumulative daily trade distribution indicated that 10% of the symbols concentrate 55% of the trade.

We assume the existence of several stock markets, with exponential symbol-interest distribution within each market. We further assume that the interest in different markets is distributed according to a Zipf distribution (i.e., $\Pr(i) \propto 1/i$), with the size of each market relative to the interest in it.

The hierarchical clustering approach was tested using the model described above. The benchmark system was comprised of 20000 symbols divided into 10 markets, and 500 users. Each user was interested in two markets, and chooses 1% of the symbols in each selected market. The average number of symbols per user was 68.85 (max=102,min=15). Figure 4 shows an example of a user interest matrix (top), and the relative message rate of each symbol (bottom), according to this model.

5.2 Estimation of Processing Costs

We estimated $\alpha(h)$ and $\beta(h)$ using the following setup. Several flows were transmitted on streams S_1 and S_2, with message rates θ_1 and θ_2. Streams S_1 and S_2, were mapped to the same multicast group. A receiver was set to accept only S_1. We assume that under medium to high load, the CPU utilization can be written as

$$\text{CPU}_{\text{Util}} \propto (\theta_1 + \theta_2) \cdot \alpha(h) + \theta_1 \cdot \beta(h). \qquad (9)$$

For a given aggregation factor, h, several combinations of the pair (θ_1,θ_2) were applied, and the CPU utilization of the receiver was measured for every combination. This allows us to estimate $\alpha(h)$ and $\beta(h)$ by mean square fit to (9). The resulting $\alpha(h)$ and $\beta(h)$ estimates were then fit with (2) and (4). Typical experimental results from a Linux system and an in-house implementation of a messaging application [12] are shown in Fig. 5. We therefor choose $h = 8$, and $\alpha(8) \cong \beta(8) \cong 1$ as typical values.

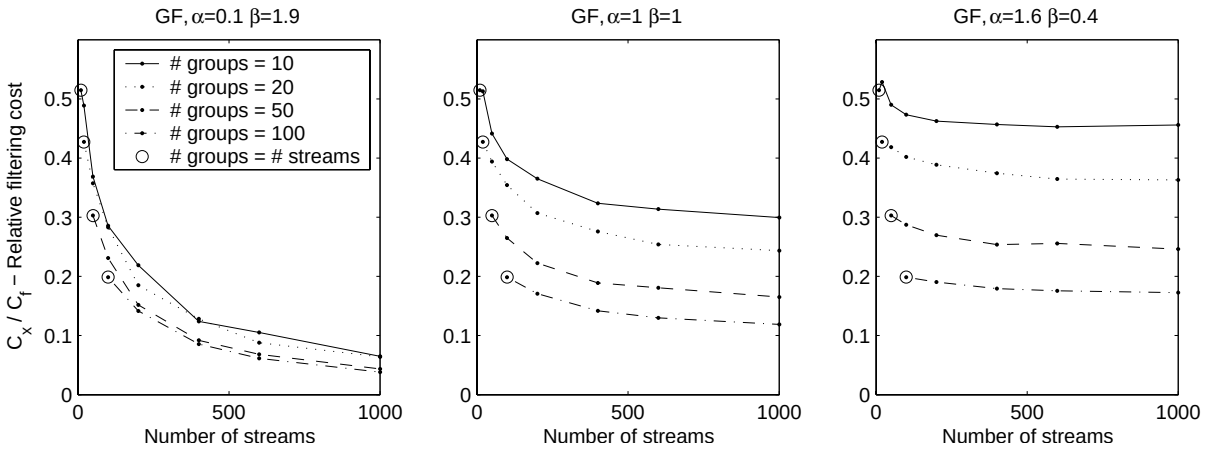

Figure 6. Hierarchic filtering cost, Two level K-mean, GF; for several values of $\{\alpha, \beta\}$, $\alpha + \beta = 2$.

5.3 Multicast Vs. Unicast and Flooding

It is instructive to compare the performance of the proposed scheme with the conventional methods of unicast and flooding. Unicast would be perfect in terms of end-node excess rate — every user receives exactly what he needs. However, given the interest matrix and message rate generated by our model (Fig. 4), a unicast transmitter would have to send each message an average of 10.1 times. Depending on the total data rate, this may be completely impractical in some systems.

Flooding means that the transmitter sends all the flows on the same stream and multicast group. This reduces the amount of multiplexing the transmitter has to do, and simplifies the system. However, the filtering load on the receivers increase. For example, given the interest matrix and message rate generated by our model, the average ratio of wanted messages to the overall messages received (goodput) would be only 2%. Let us define the flooding excess-rate filtering-cost:

$$C_{\mathrm{f}} = \sum_{n \in U} \sum_{k \in F} (\alpha(h) + \beta(h)) \cdot (I_k - w_{nk}) \cdot \lambda_k,$$

where $I_k = 1$ if $\sum_n w_{nk} > 0$, and 0 otherwise. C_{f} is used to normalize the filtering cost of the proposed scheme, presented in the following subsections.

Note that hybrid solutions are possible. For example, one could pick all the flows that have only a single interested user and transmit them by unicast. Moreover, one could also duplicate some flows on unicast links, subject to certain limitations on the number of unicast links and transmitter load. However, these options are out of the scope of this paper.

5.4 The Effect of Hierarchy

The GF algorithm was applied using a varying number of streams and groups. Fig. 6 shows C_{x}, the weighted excess-rate filtering-cost (see Sec. 3.2), normalized by the cost of flooding, C_{f}. GF results are shown for various values of α and β. In general, the results indicate that for a given number of groups, increasing the number of streams reduces the total cost; for a given number of streams, increasing the number of groups also reduces that cost. A similar pattern had been displayed by the other algorithm variants. The configuration where the number of streams is equal to the number of groups, presented as circles in Fig. 6, is equivalent to a non-hierarchical setup, as in [4, 5]. We keep $\alpha + \beta =$const. in order to maintain the cost of the non-hierarchical setup across (α, β) variations.

The relation between α and β alters the relative effectiveness of adding groups versus adding streams. When $\beta < \alpha$, as in Fig. 6-right, adding streams becomes less effective. On the other hand, when $\beta \gg \alpha$, as in Fig. 6-left, adding streams becomes almost as effective as adding groups. The typical case according to our measurements (Fig. 6-middle), is somewhere in the middle, confirming our basic approach that hierarchical clustering can significantly reduce the filtering load incurred upon the receivers, compared to the non-hierarchical setup.

5.5 Algorithm Comparison

Figure 7 shows the relative filtering cost ($C_{\mathrm{x}}/C_{\mathrm{f}}$) of the 4 variants of the hierarchical K-means algorithm, for a varying number of streams, 50 groups, and $\alpha = \beta = 1$. Results show that for a small number of streams there is hardly any difference between the variants in terms of cost. However, when a large number of streams is allowed, the GF variant shows a clear advantage over the SF variant. The IT and RRA variants shows the best performance in terms of cost, with minor differences between them (IT: 4 iterations; RRA: P_1=10, P_0=0.5, μ=0.8). The same relation between the algorithms was manifested for 10, 20 and 100 groups as well.

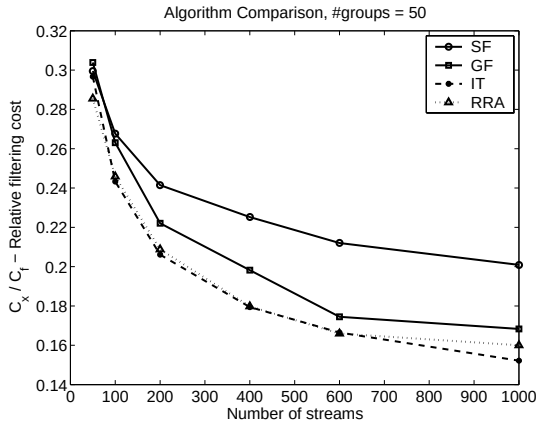

Figure 7. Relative filtering cost, SF, GF, IT and RRA; $\alpha = \beta = 1$.

6 Conclusion

In this paper we proposed a method for clustering information flows into multicast sessions, and clustering sessions into multicast groups. This approach is an evolvement of the channelization approach [3], where flows are mapped into groups directly, and sessions are not defined. This evolvement is necessitated by two factors. One factor is the large disparity between the number of flows in a large-scale data-dissemination system, and the number of practically usable multicast groups. The second factor is the observation that in such systems the filtering load incurred upon the receivers is a major performance consideration.

We formulated a novel cost function that captures the hierarchical structure of the problem, as well as the introduction of the message aggregation technique. We presented several hierarchic clustering algorithms and evaluated their performance. The receiver interest and message rate, which were used in the evaluation, were based on real-life data from the financial sector. The experimental results indicate that the hierarchical approach indeed alleviates the receiver filtering load, compared to a non-hierarchic approach. According to our perspective, this will increase the performance of the system as a whole.

We estimated the processing costs ($\alpha(h)$ and $\beta(h)$) in order to justify the additional division into sessions, since if $\beta(h) \ll \alpha(h)$ (i.e., filtering flows in the messaging layer is relatively cheap, see Fig. 3), the extra level of hierarchy does not help much. The relation between $\alpha(h)$ and $\beta(h)$ depends on implementation details, and thus it is advisable to reaffirm it on any specific system. In addition, the results demonstrate the importance of efficient stream-filtering mechanisms in the implementation of reliable multicast transport protocols.

References

[1] A. Riabov, Z. Liu, J. L. Wolf, P. S. Yu, and L. Zhang, "Clustering algorithms for content-based publication-subscription systems," in *Proc. of the 22nd Int'l Conf. on Distrib. Comp. Sys.*, Jul 2002.

[2] G. Banavar, T. Chandra, B. Mukherjee, J. Nagarajarao, R. E. Strom, and D. C. Sturman, "An efficient multicast protocol for content-based publish-subscribe systems," in *Proc. of the 19th Int'l. Conf. on Distrib. Comp. Sys.*, May 1999.

[3] M. Adler, Z. Ge, J. F. Kurose, D. Towsley, and S. Zabele, "Channelization problem in large scale data dissemination," in *Int'l Conf. on Network Protocols*, 2001, pp. 100–109.

[4] T. Wong, R. H. Katz, and S. McCanne, "A preference clustering protocol for large-scale multicast applications," in *Networked Group Communication*, 1999, pp. 1–18.

[5] ——, "An evaluation of preference clustering in large-scale multicast applications," in *Proc. of IEEE INFOCOM (2)*, 2000, pp. 451–460.

[6] B. Carmeli, G. Gershinsky, A. Harpaz, N. Naaman, H. Nelken, J. Satran, and P. Vortman, "High throuput reliable message dissemination," in *Symp. on Applied Computing*, Mar 2004, pp. 322–327.

[7] M. Oliveira, J. Crowcroft, and C. Diot, "Router level filtering for receiver interest delivery," in *Proc. of the 2nd Int'l Workshop on Networked Group Communication*, Nov 2000, pp. 141–150.

[8] T. Speakman et al., "PGM reliable transport protocol specification," RFC 3208, Dec 2001.

[9] J. Gemmell, T. Montgomery, T. Speakman, N. Bhaskar, and J. Crowcroft, "The PGM reliable multicast protocol," *IEEE Network*, vol. 17, no. 1, pp. 16–22, Jan/Feb 2003.

[10] A. K. Jain and R. C. Dubes, *Algorithms for clustering data*. Englewood Cliffs, New Jersey: Prentice Hall, 1988.

[11] "NYSE data products: National Market Volume Summary," Jul 2004, http://www.nysedata.com/info/productList.asp.

[12] "IBM Haifa Research Lab, Reliable Multicast Messaging (RMM)," http://www.haifa.il.ibm.com/projects/software/rmsdk/.