Adaptive Filtering of Visual Content in Distributed Publish/Subscribe Systems

Tarek Zaarour
Insight Centre for Data Analytics
National University of Ireland Galway (NUIG)
tarek.zaarour@insight-centre.org

Edward Curry
Insight Centre for Data Analytics
National University of Ireland Galway (NUIG)
edward.curry@insight-centre.org

***Abstract*—Classic event matching techniques in large-scale Content-based Publish/Subscribe Systems mostly rely on predicate indexing or tree-based mechanisms for fast subscription evaluation. In the context of visual analytics, such techniques are limited in supporting subscriptions requiring expensive filtering operators over unstructured event types (i.e. images and videos). In this work, user subscriptions over visual content are answered as conjunctions of commutative Boolean filters where each filter is associated with a single high-level semantic concept that may be shared across multiple subscriptions. The shared-filter ordering problem has been previously studied in centralized data stream management systems; prior works propose approximation algorithms that achieve near-optimal cost reductions in the evaluation of overlapping queries. However, in a distributed publish/subscribe setting, even an optimal ordering of filter evaluations at brokers with high workloads can create bottlenecks and waste downstream resources. We present a distributed greedy algorithm that leverages existing routing methodologies to order and distribute the execution of filters across brokers on various dissemination paths. Experiments with several pub/sub workloads show 50% to 70% decrease in event latencies and noticeable improvements in resource utilization across the overlay.**

***Index Terms*—Publish/subscribe, Expensive Filters, Shared-filter Ordering, Overlay Networks, Greedy Algorithm**

I. Introduction

Publish/Subscribe systems have been widely adopted as a loosely coupled communication paradigm for building large-scale distributed systems. Many real-world applications leverage the selective dissemination capabilities of the paradigm to exchange millions of events between geographically scattered entities. For instance, Spotify is a peer-assisted music streaming service that uses a pub/sub engine [1] for delivering hundreds of millions of user generated events every day. PubNub [2] is a proprietary network that implements a pub/sub interface to power real-time global-scale communication across many applications such as secure chat and consumer delivery tracking. Common to these applications is the need for an asynchronous and decoupled exchange of scalar events comprising textual and numerical values.

While large-scale pub/sub systems accommodate for the different filtering requirements of the above applications, they currently lack support for unstructured data types such as images and videos. Real-time multimedia applications such as traffic management, online retail, and online-gaming can benefit from the fine-grained filtering and dissemination capabilities of pub/sub systems. Nonetheless, achieving high degrees of expressiveness while maintaining the scalability of conventional distributed pub/sub systems is a challenging task especially when input events are unlabelled raw images. As an example, a subscriber interested in getting notified when a photograph of a sunny day on the beach gets published can issue a subscription to a pub/sub interface specifying the semantic concepts that the photo should satisfy (e.g. beach, sand, sunny). State-of-the-art methods in computer vision rely on learning representations using deep neural networks (DNNs) due to their high accuracy in tasks ranging from image classification to object detection. Unfortunately, it is generally acknowledged that for real-time applications with high data rates, applying DNNs is prohibitively expensive requiring extensive resources [3].

In large-scale video databases, recent works proposed preceding costly DNNs by specialized visual filters to speed up query processing [3], [4]. We observe that by embedding such filters into pub/sub broker overlays, we can offer support for visual content as native events. However, in a geographically distributed setting with a growing number of filters and subscribers, optimizing the matching process by exploiting the ordering of filter executions becomes crucial. Furthermore, subscriptions can be distributed over brokers in different locations where a routing solution would be in place to facilitate their propagation towards publisher-end brokers. Matching visual events at a single broker, even with an optimal filter ordering, can lead to centralizing the matching process, creating bottlenecks, and wasting downstream resources.

The overall contributions of this paper are:

- An approach to adaptively order and distribute expensive filters in distributed publish/subscribe systems by determining which filter(s) to execute at each broker without requiring global system knowledge.
- Extensive experiments using varying workloads in terms of the numbers of subscriptions and required filters, filter popularity distributions, and overlay topologies.

The rest of the paper is organized as follows: We discuss related works in section II, in section III we provide background on the shared-filter ordering problem. In section IV

978-1-7281-2522-0/19/31.00

we describe our publication routing graph and the adaptive filtering algorithm in detail. The experimental evaluation is provided in section V. Finally, we conclude in section VI.

II. Related Work

1) Filtering mechanisms in pub/sub: Earlier works represent events as attribute-value pairs where subscriptions comprise conjunctions of predicates over individual attributes. SIENA [5] constructs the routing table as a dictionary data structure that maintains the different Boolean predicates indexed based on the specific properties of constraint operators (i.e. =, <, >, etc.). Fabret et al. [6] cluster subscriptions based on their size and common equality predicates among other properties and perform event matching by leveraging an efficient data structure comprising a set of indexes, a predicate bit vector, and a vector of references to subscription cluster lists. Other approaches [7], [8] organize predicates in tree structures where events traverse down the tree following matching predicates towards the leaves representing different subscriptions. Since the evaluation of individual predicates is of negligible cost in the systems discussed above, approaches aim at optimizing the overall subscription evaluation efficiency as opposed to determining an optimal ordering of predicates evaluations. Determining a near-optimal filter execution order is crucial to our problem due to high filter execution costs.

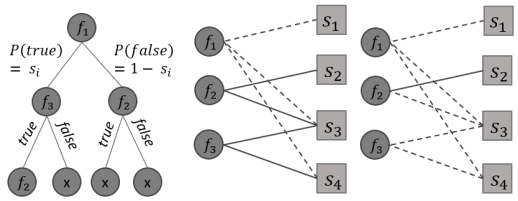
2) Expensive processing in pub/sub: There have been a few works in the literature that proposed various forms of expensive processing in pub/sub overlays. In [9] they proposed an adaptive model for supporting distributed aggregations of events represented as attribute-value pairs with the goal of minimizing the communication overhead. While in [10] they addressed customized content dissemination by embedding expensive transformation operators where clients specify the format in which they wish the data to be delivered. However, their optimizations focus on properties that are intrinsic to event aggregation or transformation.

III. Preliminaries and Problem Statement

A. Shared-filter Ordering Problem

The shared filter ordering problem consists of a set of queries Q, and a set of commutative Boolean filters F. Each query is represented as a conjunction of filters where the evaluation of each filter on an event results in a Boolean outcome of either *true* or *false*. A min-cost adaptive execution plan would result in executing a subset of all filters to determine queries that are satisfied by an event. Munagala et al. [11] identified three important factors that decide the utility of executing individual filters.

- **Cost**: c_i denotes the time cost required by a filter f_i to process an event e. Intuitively, less costly filters should be evaluated earlier.
- **Selectivity**: s_i denotes the probability that an event e will satisfy filter f_i resulting in a *true* outcome. Executing filters with low selectivity earlier would result in higher data reduction.



(a) Decision tree (b) f1: true (c) f1: false

Fig. 1: Shared-filter Optimization

- **Popularity**: p_i denotes the participation of a filter f_i in the registered subscriptions. It's intuitive that filters with high popularity should be executed earlier.

Munagala et al. [11] observed that an adaptive strategy can be conveniently represented as a decision tree (Fig. 1a) starting with an initial filter that can have two children for a *true* and *false* outcome. They propose a greedy heuristic that decides the two possible children based on a metric that combines the three factors c_i, s_i, and p_i. Suppose filter f_i occurs in p_i queries, then with probability $1 - s_i$, f_i resolves p_i queries. Hence, they define pRank of f_i as the ratio of c_i to $p_i(1 - s_i)$, where they pick the filter with the minimum pRank and recursively construct the decision tree. However, the heuristic disregards the probability s_i that filter f_i results in a *true* outcome and its impact on the current queries. The edge-coverage based algorithm presented by Liu et al. [12] overcome this limitation due to their bipartite graph representation (i.e. Residual Graph) of the query-filter mapping (Fig. 1b and 1c). An edge between two nodes in the graph means that the filter f_i is part of the conjunction of filters required by the query. If a filter evaluation outcome comes out as *true*, only the outgoing edges denoted Δ_t are covered (Fig. 1b). Otherwise, as shown in the Fig. 1c, f_1 has yielded a *false* outcome which covers all its outgoing edges and eliminates all associated subscriptions and by that also covering their outgoing edges denoted Δ_f.

$$UP(f_i) = \frac{c_i}{s_i\Delta_t + (1 - s_i)\Delta_f} \tag{1}$$

$UP(f_i)$ stands for the Unit Price of filter f_i. The filter with lower unit price will be assigned higher evaluation priority, which reflects a low cost in addition to higher expected edge cover.

B. Problem Definition

Existing approaches to filter ordering assume a single processing engine that connects all the data producers and consumers and resolves all queries registered in the system. On the contrary, in a distributed pub/sub scenario, the system comprises a federated overlay of brokers that communicate through a routing substrate (Fig. 2a). User subscriptions are routed to brokers that connect incoming event streams from various publishers, and notifications are routed through the overlay to interested subscribers. The objective of filter placement and ordering is to minimize the Processing Delay Cost which may be defined as the time required by filters executed along a

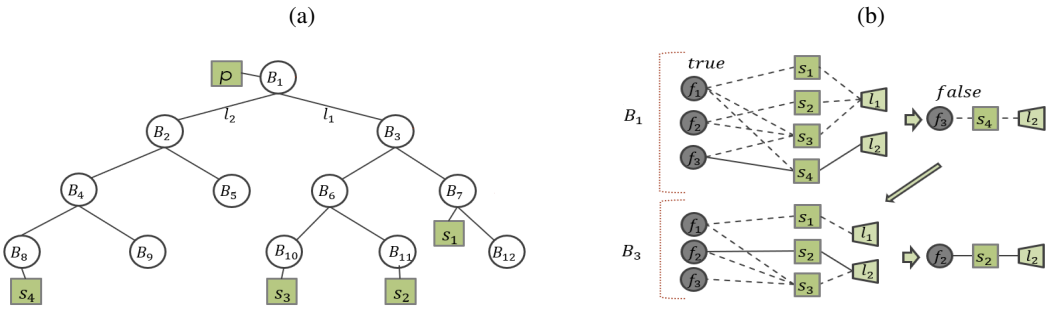


Fig. 2: (a) Federated broker overlay, (b) Message propagation schema based on the Publication Routing Graph (PRG)

path towards a subscriber to process an event. Let $B(s)$ be the set of n brokers comprising the path between a subscriber s and the corresponding publisher of interest rooted at a broker $B_j \in B(s)$. Each broker $B_i \in B(s)$ executes a subset of all required filters according to the employed execution plan. Let c_{B_i} be the cost (i.e delay) at broker B_i, then the cost of an execution plan P as perceived by s is given by:

$$cost(s, P) = \sum_{i=r}^{n} c_{B_i} \tag{2}$$

Now, we can formally define the shared-filter ordering problem in distributed publish/subscribe systems:

Definition 1: Given a set of subscriptions S, a set of filters F, and a set of brokers B. Find a shared execution plan P that minimizes $cost(S, P)$.

IV. Solution Components

A. Distributed shared-filter ordering

Selective event routing in content-based pub/sub overlays reduces message overhead through a deterministic reverse path propagation of subscriptions based on advertisement flooding. As publishers join, they issue advertisement messages that describe the type of content they intend to publish. On reception of an advertisement, a broker stores it along with the link it came from in its local Subscription Routing Table (SRT). Subscribers interested in receiving published events send their subscriptions to their connecting broker. Subscriptions are then routed along the reverse paths towards publisher-end brokers based on the SRT of each broker and are then registered in Publication Routing Tables (PRTs). Conventionally, publication matching is carried out in each broker by comparing subscription predicates against numerical or textual event attributes; on identification of a match, the notification is forwarded to the neighbouring broker where the matching process is repeated until the event reaches all interested subscribers. To this end, the introduction of visual filters departs from this match-and-forward model. More precisely, executing a portion of the required filters at a broker would enrich an image by uncovering some of the semantic concepts that it represents. Consequently, downstream brokers receiving the image need to be aware of the concepts previously identified and filters executed in order to avoid redundant processing. Our solution can be decomposed into three main components as follows.

1) Publication Routing Graph: To maintain the loosely-coupled nature of interaction in pub/sub, the ordering algorithm should only consider the local information available at each broker. Consequently, based on each broker's PRT we construct a Publication Routing Graph (PRG) as two connected bipartite graphs. Figure 2b shows the PRGs of brokers B_1 and B_3. The left-hand side of the graph represents the residual graph discussed in section III-A. The right-hand side represents a mapping between subscriptions and the links they were received through.

2) Match-and-forward Edge-Coverage-based algorithm (MEC): The MEC algorithm starts with an initial PRG at the publisher-end broker and evaluates filters one-by-one in a sequential order then terminates when all links have been resolved. At each step the decision to execute a filter is based on its unit price $UP(f_i)$ as in equation (1). The filter with the least unit price is evaluated and then removed from the graph. Based on its outcome, all satisfied subscriptions are removed as well as their outgoing edges. The unit prices of all impacted filters are then updated to select the next filter in the adaptive ordering.

Graph Short-cutting: The algorithm shortcuts the evaluation of filters based on a subscription-link mapping that helps extend the edge coverage of filters. If a link is satisfied by a matching subscription, it is removed with its associated subscriptions and their edges. Since any additional executions of filters required by subscriptions received through the same link would not provide additional information in terms of event forwarding. Consider the example in figure 2b, B_1 is aware of subscriptions s_1, s_2, and s_3 through link l_1 and s_4 through l_2. The outcome of the first filter (f_1) is *true*, hence subscription s_1 and link l_1 are satisfied. This yields the removal of s_2 and s_3, in addition to filter f_2 and by this short-cutting the execution of filters by deferring f_2 to B_3.

Extended Edge-Cover (EMEC): Prior to executing a filter f_i having probability s_i of resolving a subscription, we incorporate the possible edges that can also be covered by link elimination into $\Delta_T(f_i)$ which results in more edge coverage and decreases its unit price $UP(f_i)$. This ultimately impacts the decision of the next filter to execute leading to faster evaluation.

3) Message Propagation Schema: When the local PRG is empty, the algorithm proceeds to attach the results to the event and forwards it through matched links. When an internal broker receives an event, it first fetches the results of prior filters, updates its local PRG, and proceeds to process the event using the resulting graph. By eliminating all subscriptions received through a link as soon as a subscription that came through the same link is satisfied, we end up with two important advantages: (1) The updated PRG at the current broker is tailored towards choosing the next best filter for satisfying the remaining links without considering subscriptions that can be answered by downstream brokers. And (2) the distribution of subscriptions across brokers gets sparser as we go further down the tree, and by deferring computations downstream, the popularity distributions from the point of view of each broker will change and by that the decision will be more fitting for the subset of subscribers in each brokers local PRT.

V. Evaluation

In this section we empirically evaluate our algorithms via simulation. We implement two extreme baselines that resemble the upper and lower bounds for the shared filter ordering problem in publish/subscribe overlays. The two baselines are based on Residual Graph [12]; we implement the edge-coverage based algorithm to be executed in publisher-end brokers and subscriber-end brokers respectively. In the first case, referred to as All In Root (AIR), publisher-end brokers carry out the adaptive ordering until all subscriptions are resolved before disseminating matched events. Whereas for the second baseline, namely All In Leaves (AIL), all events are flooded towards subscriber-end brokers where the adaptive filter ordering is carried out based on local residual graphs that only consider directly connected subscribers.

A. Experimental Setup

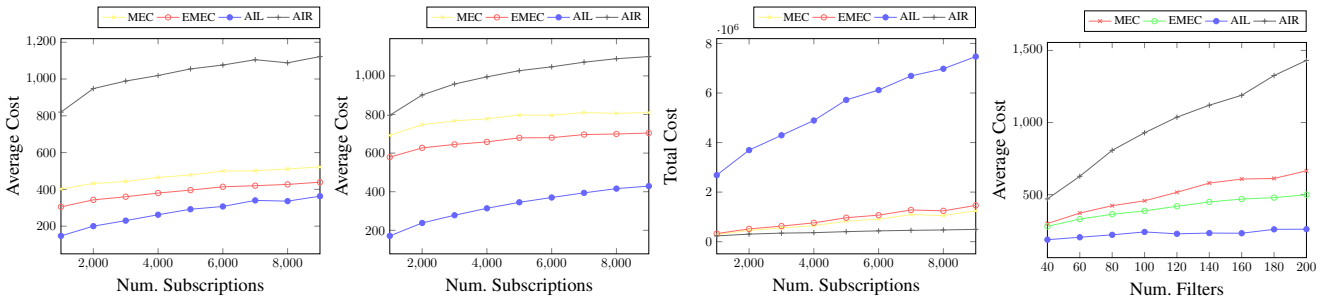
1) Overlay construction: Our overlay network implementation is based on a distributed pub/sub system simulation written in Java. We construct tree topologies based on real-world traces that provide network latencies across servers in the PlanetLab public research network. The dataset collected by Zhu et al. [13] comprises several 490×490 matrices measured over multiple time slices, where the (i, j)-th entry of a matrix m_t indicates the measured Round Trip Time (RTT) from node i to node j at time t. We select 100 nodes at random to host the brokers and compute the Minimum-Spanning-Tree (MST) that connects them. We follow this process to create five different pub/sub overlay topologies.

2) Instance generation: Based on the work by Mungala et al. [11], the filters are generated by specifying their cost and selectivity values based on a random distribution from the pre-set value ranges. We specify the costs of individual filters out of the range $[12, 16]$ that was decided via extensive prototyping using MobileNets [14]. An efficient CNN architecture that achieves comparable accuracy to deep models while surfacing two hyper-parameters that can be tuned to build very small, low latency models. The selectivity range was set to $[0, 0.25]$ because the matching ratio of subscriptions is known to be low [9] in pub/sub literature. We then generate subscriptions comprising combinations of filters selected following a Zipf Distribution due the close resemblance between image filtering based on object category occurrences and text search engines and the inverse relationship between the frequency of word occurrence and its rank. Finally, we generate an event stream of consecutive event items that are simulated by deciding for each event whether each filter f_i has a *true* or *false* outcome based on its selectivity s_i. Subscriptions are distributed across brokers based on a load-value $\beta \in [0, 1]$ chosen randomly for each broker. For each run, the publisher issues 1000 events; each experiment instance is repeated 5 times using 5 different typology structures resulting in a total of 25 runs for each configuration.

B. Experiments

1) Impact of subscriptions: In the first experiment we vary the number of subscriptions while fixing the number of filters to 100. We measure the average processing delay cost along every path towards subscribers which accounts for the impact of the processing cost at each broker (i.e. filter executions) on the average perceived latency by subscribers while excluding communication costs. As shown in Figure 3a, the MEC and EMEC algorithms perform $50 - 70\%$ better than the AIR baseline when filter popularities follow a Zipfian distribution. The sharing degree in this experiment is highly skewed, which makes a few filters very popular and leads to resolving many subscriptions at an early stage by adaptively prioritizing filters with high edge-coverage. Figure 3b shows the same experiment under a uniform distribution where we eliminate skew by randomly assigning every filter a weight between [0,1] and assigning filters to subscriptions based on a weighted probability. Results show a noticeable increase in the processing delay cost of all approaches because the adaptive ordering of filters requires more filter executions to resolve all subscriptions when the popularity of filters is less skewed. Furthermore, EMEC consistently outperforms all other approaches due to short-cutting the residual graph earlier by incorporating the extended edge cover into the calculation of a filter's unit price.

2) Redundancy overhead: In the third experiment, we measure the redundancy overhead of all approaches as the total cost of filter executions across all broker in the overlay. Results in Figure 3c show how AIL leads to a linearly increasing total cost as the number of subscriptions increases. This is because sharing across subscriptions is only leveraged locally at each subscriber-edge broker as opposed to sharing filters early on in the overlay. We can also see how AIR introduces the least overhead due to executing each filter only once at the publisher-end broker. Even though our algorithms can lead to some redundancy by executing the same filters at different paths due to offloading, this is balanced by decreasing the number of filters executed on each path, which explains the slight increase in processing cost as shown in Fig. 3c. We also highlight how EMEC consistently leads to slightly more



(a) Zipfian Distribution (b) Uniform Distribution (c) Redundancy Overhead (d) Varying filters

Fig. 3: Experimental Results

total cost than MEC because of offloading more filters to be executed downstream by eliminating more subscriptions earlier with the extended edge cover resulting in more redundant processing on separate paths.

3) Impact of filters: In this experiment we fix the number of subscriptions to 3000 and vary the number of filters. We observe that the performance of AIR is more sensitive to the increase in the number of filters as compared to that of subscriptions. This is because in the latter, all algorithms use the same set of filters in the evaluation process. As the number of subscriptions increases, each filter will have a larger chance of being evaluated but the overall cost is bounded by the total cost of all filters available in the system. Figure 3d shows how the processing delay cost when applying MEC or EMEC degrades slower due to distributing the execution of filters.

VI. Conclusion

In this work, we tackle the problem of supporting visual content filtering in distributed publish/subscribe systems. We propose the use of pub/sub as a medium for an asynchronous and decoupled interaction between multimedia producers and consumers. More precisely, we introduce fast binary filters into a pub/sub overlay of dedicated brokers, we propose a distributed and adaptive greedy algorithm to order and distribute the execution of filters across paths towards subscribers. Our experiments with varying pub/sub workloads show clear improvements in the average processing delay cost perceived by individual subscribers while keeping message overhead to a minimum. An interesting avenue for future work would be to look at filters that are trained to detect multiple concepts instead of binary filters. This increases the complexity of the ordering problem where each individual filter can hold the impact of different concepts on the current registered subscriptions.

Acknowledgment

This publication has emanated from research conducted with the financial support of Science Foundation Ireland (SFI) under Grant Number SFI/12/RC/2289_P2, co-funded by the European Regional Development Fund and the European Union's Horizon 2020 research programme Transforming Transport (TT) grant No 731932.

References

[1] V. Setty, G. Kreitz, R. Vitenberg, M. Van Steen, G. Urdaneta, and S. Gimåker, "The hidden pub/sub of spotify:(industry article)," in *Proceedings of the 7th ACM international conference on Distributed event-based systems*, pp. 231–240, ACM, 2013.

[2] "Pub/sub messaging." https://www.pubnub.com/.

[3] D. Kang, J. Emmons, F. Abuzaid, P. Bailis, and M. Zaharia, "Noscope: optimizing neural network queries over video at scale," *Proceedings of the VLDB Endowment*, vol. 10, no. 11, pp. 1586–1597, 2017.

[4] Y. Lu, A. Chowdhery, S. Kandula, and S. Chaudhuri, "Accelerating machine learning inference with probabilistic predicates," in *Proceedings of the 2018 International Conference on Management of Data*, pp. 1493–1508, ACM, 2018.

[5] A. Carzaniga, D. S. Rosenblum, and A. L. Wolf, "Design and evaluation of a wide-area event notification service," *ACM Transactions on Computer Systems (TOCS)*, vol. 19, no. 3, pp. 332–383, 2001.

[6] F. Fabret, H. A. Jacobsen, F. Llirbat, J. Pereira, K. A. Ross, and D. Shasha, "Filtering algorithms and implementation for very fast publish/subscribe systems," *ACM Sigmod Record*, vol. 30, no. 2, pp. 115–126, 2001.

[7] S. Kale, E. Hazan, F. Cao, and J. P. Singh, "Analysis and algorithms for content-based event matching," in *Distributed Computing Systems Workshops, 2005. 25th IEEE International Conference on*, pp. 363–369, IEEE, 2005.

[8] M. K. Aguilera, R. E. Strom, D. C. Sturman, M. Astley, and T. D. Chandra, "Matching events in a content-based subscription system," in *Proceedings of the eighteenth annual ACM symposium on Principles of distributed computing*, pp. 53–61, ACM, 1999.

[9] N. K. Pandey, K. Zhang, S. Weiss, H.-A. Jacobsen, and R. Vitenberg, "Distributed event aggregation for content-based publish/subscribe systems," in *Proceedings of the 8th ACM International Conference on Distributed Event-Based Systems*, pp. 95–106, ACM, 2014.

[10] H. Jafarpour, B. Hore, S. Mehrotra, and N. Venkatasubramanian, "Ccd: A distributed publish/subscribe framework for rich content formats," *IEEE Transactions on Parallel and Distributed Systems*, vol. 23, no. 5, pp. 844–852, 2012.

[11] K. Munagala, U. Srivastava, and J. Widom, "Optimization of continuous queries with shared expensive filters," in *Proceedings of the twenty-sixth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pp. 215–224, ACM, 2007.

[12] Z. Liu, S. Parthasarathy, A. Ranganathan, and H. Yang, "Near-optimal algorithms for shared filter evaluation in data stream systems," in *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pp. 133–146, ACM, 2008.

[13] R. Zhu, B. Liu, D. Niu, Z. Li, and H. V. Zhao, "Network latency estimation for personal devices: A matrix completion approach," *IEEE/ACM Transactions on Networking*, vol. 25, no. 2, pp. 724–737, 2017.

[14] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.