P3: A Privacy Preserving Personalization Middleware for recommendation-based services

Animesh Nandi, Armen Aghasaryan, Makram Bouzid
{Animesh.Nandi, Armen.Aghasaryan, Makram.Bouzid}@alcatel-lucent.com

Bell Labs Research, Alcatel-Lucent

Abstract

We propose the design of a privacy-preserving-personalization middleware that enables the end-user to avail of personalized services without disclosing sensitive profile information to the content/service-provider or any third party for that matter. Our solution relies on a distributed infrastructure comprising local clients running on end-user devices and a set of middleware nodes that could be collaboratively donated by few end-users or hosted by multiple non-colluding third parties.

The key idea is to locally compute the user's profile on the device, locally determine the interest group of the user wherein an interest-group will comprise users with similar interest, and anonymously aggregate the collective behaviour of the members of the interest group at some middleware node to generate recommendations for the group members. In addition, our system is also open for third party content and recommendation injection without leaking the users privacy.

1 Introduction

A broad class of applications such as StumbleUpon [25] (or iGoogle [18]) URL recommendations, Foursquare [14] check-in recommendations, Netflix [21] movie recommendations, or IPTV content recommender systems suffer from the dilemma of having the user disclose sensitive profile information in order to benefit from personalized content/services. As of today, users have no option but to trust the content/service-provider with their sensitive profile information in return of the personalized content/services they seek.

Typical centralized recommender systems rely on one of the two main types of recommendations: content-based and collaborative filtering. The general content-based technique exploits content metadata (categories and tags) and based on the user's content consumption history builds a profile of the user in terms of weights associated with different categories and tags. Having built the profile of a user, items whose content metadata matches the tags and categories that have high weights are recommended. The general collaborative filtering procedure for content recommendation can be described as follows: 1/ compute clusters of similar users based on their history, 2/ compute the popularity curve of items within each cluster, and 3/ recommend to each user the most popular items within his cluster that are not present in his own history. As compared to content-based recommendations, collaborative filtering enable users to discover new types of content they may like and that they never consumed before nor expressed any explicit interest in them. However these approaches require having access to the users' profiles to take advantage from each other's experiences.

We would ideally like to have a privacy preserving personalization system that enables the end-user to benefit from a personalization/recommendation service without disclosing their preferences (i.e. user profile) to the content/service provider. Although there has been some prior work towards this goal, prior systems suffer from several limitations (as detailed in Section4) which prevent them from being deployed globally and adopted widely by the end-users.

We will now describe some desired properties of an ideal privacy-preserving-personalization system.

- **Generic middleware for wide range of applications**: The solution should support wide gamut of potential applications varying from Telco-based IPTV content personalization, Web-based personalization like iGoogle URL recommendations, and mobile-based LBS (Locality-based services) personalization like Foursquare check-in recommendations etc. Thereby, the design should be in the form of a middleware, where different recommendation applications potentially running on different types of end-user devices (e.g. mobile phones, PCs, or Set-Top-Box) can plug-in.

- **Hybrid recommender system**: Given the obvious advantages of both content-based and collaborative filtering recommendations, we target a hybrid recommender system which benefits from both content semantics/metadata (content-based approach) and from the experience of other users (collaborative filtering approach).

- **Seamless Operation of Content-providers**: The design should enable 'seamless' (i.e. without requiring explicit cooperation) interaction with external content/service providers to receive appropriate recommendations generated by proprietary recommendation algorithms that have be developed and fine-tuned by content/service-providers like for example Google. Thereby, the ideal system should not require the content-provider to change fundamentally its APIs/interfaces, and additionally can work without having to invest in replicating sophisticated state-of-art recommendation algorithms developed by content-providers.

- **Profile Anonymization with 'Unlinkability'**: The design should obviously break the association of the user from his profile. In addition, even if the profile is anonymized, no one node should be able to see the complete profile, which we claim is prone to sophisticated linkability attacks wherein one can infer the mapping of pseudonym to user based on the profile details. Therefore we strive for unlinkability as well, which is achieved by ensuring that any single node can see only a small slice of the entire profile.

- **Trust No One**: Our design is aimed at ensuring that we do no trust any single-entity with all our sensitive information. Secondly, our system should be designed to be able to work under small scale collusion attacks.

- **Comparable Performance**: Last but not the least, the system should provide recommendation quality that is comparable or only slightly inferior to that of centralized recommender systems, at the cost of slight increase in overheads like communication costs or infrastructure costs.

We will next describe the design of our proposed P3 (i.e. acronym for privacy-preserving-personalization) system which is intended to meet the design goals mentioned above.

The rest of the paper is outlined as follows - Section 2 describes the design of the P3 system, Section 3 describes the detailed realization of the design, Section 4 contrasts our P3 system with prior works, and finally we conclude in the Section 5.

2 Design of the P3 Architecture

In this section we will highlight the key contribution of the work in terms of showing how different *functional blocks* can be interconnected to meet the goals of the privacy-preserving-personalization architecture.

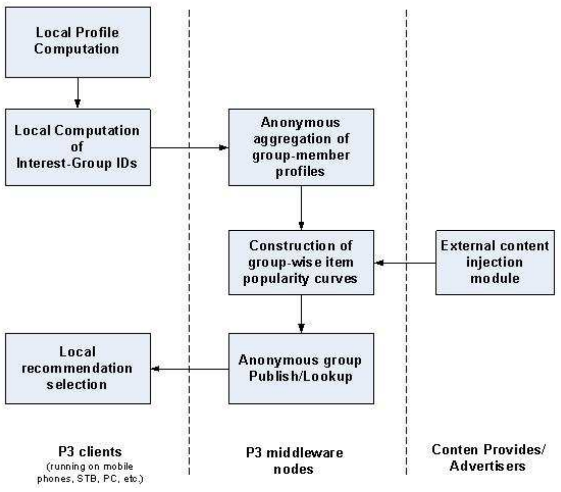
Local Profile Computation
Local Computation of Interest-Group IDs
Anonymous aggregation of group-member profiles
Construction of group-wise item popularity curves
External content injection module
Local recommendation selection
Anonymous group Publish/Lookup
P3 clients (running on mobile phones, STB, PC, etc.)
P3 middleware nodes
Conten Provides/ Advertisers

Figure 1: Functional Blocks of our Privacy-Preserving-Personalization Middleware

In order to protect the user privacy we propose to execute the steps of a centralized recommender system in a privacy-preserving way by using a distributed infrastructure comprising of *P3 local clients* running on end-user devices and a *set of P3 middleware nodes* hosted by multiple non-colluding entities. Models like that of Virtual Individual Servers (VIS) [6] or TOR [13] which employ end-users running nodes locally or in the cloud computing platforms (e.g. Amazon) could be envisioned. Note that we only require a small fraction of users to act as volunteers to host these nodes, and incentive mechanisms can be provided to such users. Alternatively, one could also imagine multiple third-parties like different cloud computing providers (e.g. Amazon, Google, Microsoft) together hosting the P3 middleware nodes.

The key idea is to build users' profiles locally on their personal devices, to identify the interest-group that a user is associated with locally wherein an interest-group will comprise users with similar interest, and anonymously aggregate the collective behaviour of the members of the interest-group to do both types of recommendation - collaborative filtering as well as content-based.

The core contribution of the work revolves around the idea of how the different functional blocks described in Figure 1 can be combined in order to realize the goals of our privacy-preserving-personalization middleware. Although our proposed realizations of each functional block as described in Section 3 exist in some adapted form in prior state-of-art, the key contribution of our work lies in how the interconnection of the functional blocks enables us to realize our desired goal. While referring to Figure 1, we will now describe the role of each functional block and how they interconnect.

STEP 1 ('Local Profile Computation' block in Figure 1): First, each user client collects the local traces of users' activities in the targeted service domain. These traces are analyzed and compacted in such a way that they represent the user profile, i.e. the user preferences in terms of items or in terms of item categories. Such a local profile can for example be represented as a set of <key, value> pairs, as it is described in Section 3. The P3 middleware can provide the personalization service to any external application for which such a local profile is available.

STEP 2 ('Local Computation of Interest Group IDs' block in Figure 1): Then, each local client determines its user's interest group by considering only the local profile data and some globally available information (e.g. a concept taxonomy, a term vocabulary, or seeds for generating random vectors). Note that this can be done without sending out the local profile to any external entity by using techniques like LSH (Local Sensitivity Hashing) [16] as described in the Section 3.

STEP 3 ('Anonymous aggregation of group member profiles' block in Figure 1): At the next step, we aggregate anonymously all the group member profiles corresponding to a particular interest group at a dedicated middleware node which we call the *Group-wise aggregator*. All the data related to a given group are collected in a single Group-wise aggregator.

STEP 4 ('Construction of group-wise item popularity curves' in Figure 1): Using collected group members' profiles, the Group-wise aggregator will generate a set of potential recommendations for the interest group based on the aggregate statistics of the already consumed content by the group members. In particular, the Group-wise aggregator can compute the top-K popular items of the group and regard these as the collaborative-filtering recommendations for the group.

STEP 5 ('External content injection module' in Figure 1): The Group-wise aggregator can communicate the categories/tags corresponding to the top-K items to the external content provider to get content-based recommendations corresponding to the interests of the group. In fact, it can also seamlessly interact with the external content provider by posing as an end-user who consumes the top-K popular items of the interest group and get recommendations generated as per the proprietary recommendation algorithm running at the content provider.

STEP 6 ('Anonymous Publish/Lookup of group recommendations' in Figure 1): The generated recommendations at the Group-wise aggregator, which are a combination of the internal collaborative-filtering recommendations and the external recommendations, can be looked up anonymously by the group members or alternatively published to the group members anonymously.

STEP 7 ('Local recommendation selection' in Figure 1): Finally, the list of recommendations is filtered out on each user device, by removing already consumed content, and used to make recommendations for the end user.

3 Realization of the P3 Functional Blocks

Bellow we give more details on the realization of the functional blocks introduced in the previous section. We map these functions on different types of components of a distributed system which host the P3 middleware. Figure 2 describes one possible realization of the functional blocks.

3.1 Local profile computation

Component type: P3 local client running on end-user's device

We consider two possible realizations for this phase; of closed and open systems (in terms of consumed content space) respectively. We describe how these types of systems or applications allow the construction of local profiles in terms of <key, value> pairs, where keys are either items (item references) or item categories (tags, taxonomy concepts, etc.) while the values represent the interest level (e.g. on a 0 to 1 scale).

(a) Example of <key, value> construction in a closed-system: content provider portal, e.g. VoD portal. Each content item is explicitly associated with its metadata provided by the content provider. These metadata include the title and/or the artists and/or genres and/or keywords/tags, etc. The user consumptions are then mapped to these metadata terms and reflect user interests; each user consumption itself is a set of <key, value> elements where key is the metadata term and value is the interest towards that term. The aggregation of these consumption sets over the time allows to infer user interests in the form of <key, value> elements [1]. In this case, no additional global information is necessary for local profiling other than the metadata provided by the portal.

(b) Example of <key, value> construction in an open system: web browsing. In this case, the content items are web pages and their unique identifiers are the URLs. In addition, each web page visited by the user can be processed to find additional metadata characterizing its content. Such metadata can be either specified explicitly under the HTML 'title' tag or meta tags 'keyword' and 'description', or discovered implicitly, by parsing the source text of the web page content. In any case, after the extraction of these metadata an additional normalization is needed to incorporate some readjustments wherein common tags are given lower rates akin to IDF (Inverse Document Frequency) analogy. The normalization procedure can be based on some global tag popularity that can be retrieved for instance from Google Analytics.

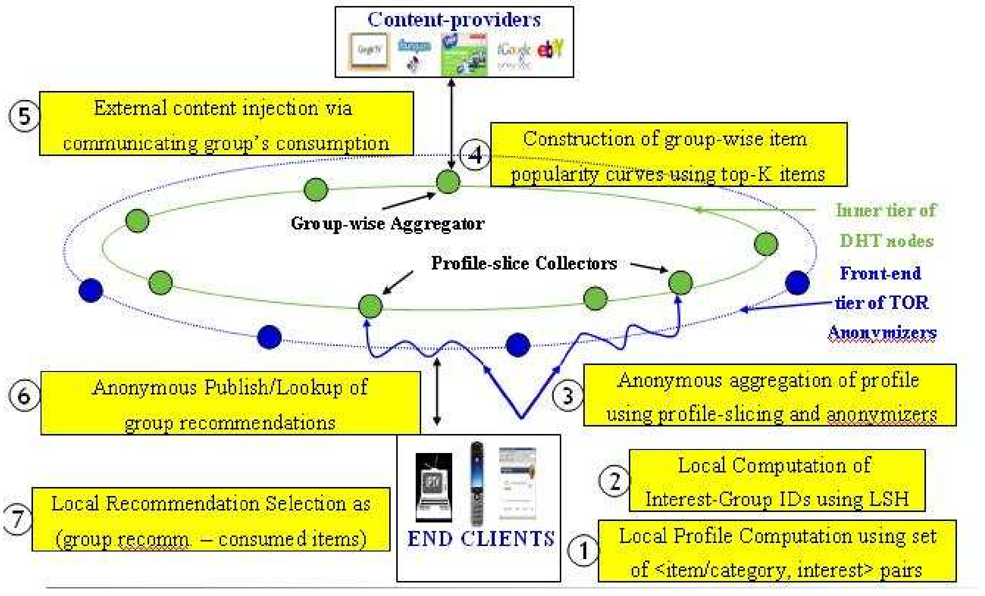
Content-providers
5
External content injection via communicating group's consumption
4
Construction of group-wise item popularity curves using top-K items
Group-wise Aggregator
Inner tier of DHT nodes
Profile-slice Collectors
Front-end tier of TOR Anonymizers
6
Anonymous Publish/Lookup of group recommendations
3
Anonymous aggregation of profile using profile-slicing and anonymizers
2
Local Computation of Interest-Group IDs using LSH
7
Local Recommendation Selection as (group recomm. – consumed items)
END CLIENTS
1
Local Profile Computation using set of <item/category, interest> pairs

Figure 2: A realization of the functional blocks on different components of the P3 Middleware

Note that in both examples keys can represent unique identifiers such as the content title as well as other types of metadata such as genres or tags which characterize more than one content. The advantage of mixing both unique identifiers and categories is that the framework allows applying hybrid types of recommendation algorithms using both collaborative filtering (exploiting only the unique item identifiers) and content-based recommendations (exploiting the content categories). Optionally, to avoid mixing different types of keys, the <key, value> data structure can be replaced by <item-category, item-list, value> where item-category stands for keys that represent categories and item-list stands for key that represent unique identifiers content items the consumption of which contributed to inferring user interest in the given item-category.

Note that in both cases of closed and open systems keys that represent item categories can be mapped onto a common ontology, a taxonomy or flat vocabulary terms. This will also ensure semantic uniformity between different local profiles and also between different applications requesting the personalization service from the P3 middleware [2]. The mapping dictionaries then should be available locally at each local client. Note that such a semantic uniformity facilitates the local clustering described in the next session.

3.2 Local computation of global Interest-Group IDs

Component type: P3 local client running on end-user's device

The objective of this phase is the local computation of user interest groups (clusters), i.e. each local client determines its user's interest group by considering only the local profile data and some globally available information (e.g. a concept taxonomy, a term vocabulary, or seeds for generating identical random vectors). The global information is shared among all the clients independently of their local profile content; by mapping the local profiles onto this global information space one can determine the interest-groups as clusters characterized by commonly shared terms. So, two distinct user profiles that map (locally) to the same cluster discriminator will make part of the same interest-group without the need for explicit pair wise comparison between these two local data entities.

Two examples of such local interest-group computation can be given.

(a) Example using Local Sensitivity Hashing (LSH) techniques: an approach to compute all the similarities between all users in a computationally efficient way is given by LSH techniques [16]. This approach is commonly used to solve nearest neighbor and clustering problems in case of high dimensional spaces, where the 'Curse of dimensionality' makes an exhaustive search unfeasible [1]. The basic idea behind LSH is that two similar objects will hash to the same value with high probability, the value outputted by the hash functions could be used as the 'label' of a cluster (or cluster_id) to which the two objects belong. The output of several hash functions could be concatenated in order to reduce the probability of errors.

Google is using MapReduce and LSH to run a collaborative filtering algorithm that generates recommendations for users of Google News [12]. Here, we suggest to apply a similar approach in order to compute user similarities by using only local processing at each end-user client.

Each user client executes the appropriate LSH algorithm depending on the chosen distance metrics, e.g. Charikar offers a solution to implement cosine similarity using LSH [8], and similarly can use the MinHash algorithm [9, 10] to implement set similarity using the Jaccard coefficient.

The LSH code obtained from the local profile data gives the cluster identifier(s), cluster_id, which are statistically identical for similar users. One can generate several LSH codes and concatenate them for the same user to reduce false positives. At the same time, in order to catch several aspects of user's consumption patterns we can hash each user to a number of clusters (as suggested in [12]).

Finally, note that in an open system described in the previous section, the LSH will be computed by using only the keys representing item-categories. Note that privacy is not compromised in this process as all the computation is done locally.

(b) Example using semantics-based clustering: Each local client can identify the top-K item-categories within the local profile. This list will be considered as cluster descriptor of the interest group to which the given user belongs to. As an extension to this approach, one can also consider different subsets of top-K categories of size m, $m < K$, so that the given user is affiliated to more than one cluster.

3.3 Anonymous aggregation of group-member profiles

Component type: P3 Middleware nodes

To compute the item popularity curve within each interest group, one needs to collect in a single place all the local profiles belonging to the same cluster. Doing this directly would however violate the user privacy because the local profile of each member user could be distinguished during the data transfer. To allow anonymous aggregation of profiles, we divide this process in two steps: (1) collecting partial profile data via an anonymization network like TOR [13] (i.e. Onion Routers [24]) at arbitrarily chosen P3 middleware nodes (called Profile-slice collectors) that participate in a DHT [11] (i.e. Distributed-Hash-Table), and (2) aggregating the information on a per cluster basis at the P3 middleware node (called Group-wise aggregator) which is the DHT node responsible for storing/computing information related to the corresponding cluster. Such a DHT node is chosen using the DHTs key-based-routing (KBR) primitive with the cluster_id as the key.

The role of the anonymization network is to hide the identity of the end-users from the Profile-slice collectors. The goal of slicing the profile into small segments is to ensure 'unlinkability' such that even after anonymization, no middleware node sees enough of the profile to be able to intelligently infer the identity of the profile owner from the profile contents.

Below we provide more details on the two-step aggregation mechanism.

(1) Profile slice collection: At this step, each user client slices the local profile into segments $s_1, , s_n$ composed of one or more <key, value> elements each. The profile slicing mechanism should be intelligent to ensure that no profile-segment by itself contains enough profile content

[1]http://en.wikipedia.org/wiki/Curse_of_dimensionality

items that can be pieced together to infer the identity of the individual even after anonymization. By slicing to small-enough segments (with the limit being single item per segment), the probability of intelligent inference attacks on a profile segment reduces. However, one can design a profile slicing mechanism that minimizes the number of segments to reduce communication overhead while still ensuring safeguard from intelligent inference attacks.

Then, each segment together with the users cluster identifier(s), i.e. $< s_i$, cluster_id> are sent via an anonymization network like TOR to different profile slice collectors. A typical 3-hop onion-routing path [24] can be established with the end-user encrypting the $< s_i$, cluster_id> information with the public-key of the exit node of the onion-routing path, and the exit-node decrypting the information and relaying it to the profile slice collector. The end-user periodically chooses a different random set of DHT nodes to act as its set of Profile-slice collectors. Thus, different parts of the local profile will be delivered to different profile slice collectors, and none of them will have a full view of the local profile of any user.

Its worth noting that in scenarios where the end-user clients device is well-provisioned (e.g. IPTV set-top-box) to be a member of the DHT itself, the TOR-based anonymization network could be optionally replaced by a DHT-based anonymization mechanism like AP3 [20]. In this case, the client nodes generate a new hash code H_i for each pair $< s_i$, cluster_id> and select a DHT node as a profile slice collector using H_i as the DHTs KBR key, and send the profile-slice anonymously using AP3 to this profile slice collector.

(2) Group-wise aggregation: At this step, each profile slice collector forwards a given data element $< s_i$, cluster_id> to a Group-wise aggregator selected with a DHTs KBR mechanism by using the cluster_id as the hash code or key. As a result, all the data elements related to a given cluster will be collected in the same physical node. Note that each DHT node could be responsible for several clusters; this will depend on the number of clusters generated and the total number of DHT nodes.

Its worth mentioning here that one can explore other complex alternatives for anonymous aggregation like Anonygator [23] which in addition to some of the desired properties mentioned above can give additional protocol properties like resistance to data pollution by malicious end-users and more scalable aggregation using multi-trees. Similarly, a system like that proposed by Applebaum et al [3] doing cryptographic-based aggregation of <key, value> pairs, can be used for stronger privacy guarantees under collusion attacks as well as additional properties like 'keyword privacy' where we can ensure that the Group-wise aggregator can only know the top-K items of the interest group instead of the entire group's consumption.

3.4 Construction of group-wise item popularity curves

Component type: Group-wise aggregator

The group aggregator responsible for the given cluster concocts all the <key, value> elements belonging to its members. This allows computing the item popularity curve of each cluster, or in other words, each recommendation peer will select the top-K items for the given interest group. The top-K items could either be items, categories or tags. These top-K items serve as the group recommendations generated by the collaborative filtering approach. Additionally, they also reflect the primary interests of the group members in terms of items, categories or tags.

3.5 External content injection

Component type: External content injection module

This module is optional; it connects group-wise aggregators to external content and ad providers to push external content sources like targeted ads or third-party content recommendations. To achieve this functionality, the group-wise aggregators can either explicitly pull recommendations on behalf of its group; or alternatively can interact 'seamlessly' with the external providers without requiring any special cooperation from the external provider.

In the first type of interaction (i.e. explicit), the group-wise aggregators can communicate the primary interest of the group in terms of categories and tags corresponding to the top-K items of the interest group to the external content provider to get content-based recommendations. The latter can return a list of contents/ads to be injected in the respective Top-K list. This external module will make its suggestions in correspondence with the interest group.

In the second type of interaction (i.e. seamless), the group-wise aggregator can seamlessly interact with the external content provider by posing as an end-user who consumes the top-K popular items of the interest group or for that matter the entire consumption list of its group members as its own consumption. The external content provider profiles the group-wise aggregator just as it profiles any other end-user, and generates recommendations (by any type of specific recommendation algorithm) for the group-wise aggregator, which in reality represents the interests of the group members.

3.6 Anonymous group Publish and Lookup of group recommendations

Component type: P3 Middleware nodes

The recommendations generated at the group-wise aggregator can be made available to the group members either via a PULL-based approach (i.e. lookup) or via a PUSH-based approach (i.e. publish). As in the aggregation mechanism, such a lookup or publish must be anonymous as well. We will describe two realizations below via the anonymous lookup and the anonymous publish-subscribe mechanism.

(a) Anonymous Lookup: In this operation, the end-user's P3 client issues a DHT-lookup [11] by using the cluster_id as the hash code or key. However such a DHT lookup is done over a typical 3-hop onion-routing path [24], with the cluster_id encrypted with the public-key of the exit node of the onion-routing path, and the exit-node decrypting the cluster_id, issuing a DHT-lookup with cluster_id as the DHTs KBR key, encrypting the returned results of the DHT-lookup (i.e. the recommendations) with the symmetric key that is provided by the end-user. The encrypted recommendations are sent back on the reverse path and the end-user's P3 client finally decrypting the recommendations. Note that privacy is preserved in the lookup mechanism, because the only node that knows the identity of the end-user is the entry-node of the onion-routing path, which however can neither know the end-user's group association (i.e. due to encrypted cluster_id) or the group's recommendations (encrypted recommendations). Its worth noting that in scenarios where the end-user client's device is well-provisioned (e.g. IPTV set-top-box) to be a member of the DHT itself, the TOR-based anonymization network could be optionally replaced by DHT-based anonymization system using AP3 [20].

(b) Anonymous Publish: Alternative to the above PULL-based anonymous lookup mechanism, one can employ a more complex PUSH-based anonymous publish-subscribe mechanism, wherein newly generated recommendations at the group-wise aggregator are published to the group members anonymously, i.e. ensuring that no P3 middleware node including the group-wise aggregator knows the identity of the group members. This can be done using anonymous channels that allow an end-user to specify a kind of mailbox-address for its intended messages as the channel address without divulging their identity. When a node wishes to construct an anonymous channel, it first picks a random id, as the address of the channel. Messages sent to this channel id are then forwarded anonymously back to the receiver, and nodes who send messages to the channel are unaware who is the actual recipient. Thus, if an end-user wishes to join the publish-subscribe group corresponding to cluster_id, it first creates an anonymous channel and then includes the address of the channel in the anonymously routed group-subscription request to the group-wise aggregator. The channel ids corresponding to different group members that the group-wise aggregator receives is maintained as the group state. Generated recommendations at

the group-wise aggregator can then be sent to the channel ids corresponding to its group members. Both TOR [13] as well as AP3 [20] allow the creation of anonymous channels. As in the case of anonymous lookups, depending on whether the end-user's P3 client is provisioned enough to run a DHT node locally, we use either an AP3 or TOR based realization of anonymous channels.

3.7 Local recommendation selection

Component type: P3 local client running on end-user's device

On the reception of group recommendations corresponding to the user's interest group(s), the user's client compares them with the local profile; by making a simple extraction operation one can filter out the items already consumed by the user in the past and obtain the recommendation list from merging the remaining items in the top-K lists provided by each cluster the user in affiliated to. In the case, where targeted ads have been injected in top-K lists a specific behaviour can be adopted by the user client depending on the user's preferences. On two extremes, all the ads could be filtered out or all the ads could be kept in the list which will allow repeated visualization of the same ad over the time).

4 Related Work

In this section, we will categorize the related work according to high level functionalities provided by prior state-of-art, and contrast their functionality with that of our P3 system.

Privacy preserving local content-based recommender systems: A few works like Adnostic [26] and Privad [17] employ a local profiling algorithm that categorizes the user into audience segment(s) based on the observed browsing and content consumption history. The content-provider broadcasts all ads/content to each user, together with specification of the intended audience-segment corresponding to each ad/content. The local content-filtering algorithm however only displays ads/content corresponding to the user's audience-segment.

To reduce the bandwidth costs resulting from broadcast of all content/ads, systems like Privad [17] anonymously subscribe the user to high-level interest categories via an anonymizing proxy that is assumed to not collude with the content-provider, receive content/ads for this high level category, and do fine-filtering of relevant content/ads locally.

Such systems however face the challenge of being able to develop an algorithm that can run on end-user's device with recommendation quality comparable to proprietary recommendation algorithms developed by centralized content-providers like Google etc. In addition, these systems only provide content-based recommendations and are unable to provide collaborative filtering recommendations since there is no way to know other top-rated items liked by users with similar interests.

In contrast, P3 enables both content-based as well as collaborative filtering recommendations, is more scalable, has greater resistance against collusion attacks, and also does not have to rely on replicating proprietary personalization algorithms on the local device.

Another very recent work Re-Priv [15] relies on doing local profiling in the browser. However, after having done the local profiling it relies on users to give explicit permissions to service/content providers to utilize the meta-profile computed locally to push personalized content. Their privacy benefits come from the fact that the browsed urls need not be revealed but only the synthesised meta interests can be provided to the content provider. In contrary, our approach is aimed to even hide the high level interests from the service/content provider.

Privacy-preserving collaborative filtering: Collaborative Filtering (CF) approaches present big privacy issues since they need to gather all users consumptions on a centralized server. Anonymizing users' profile (i.e via assigning pseudonyms) before sending them to a centralized server is prone to 'linkability' attacks where the real identity of the pseudonym can be inferred based on complete profile details (e.g. AOL scandal in 2006).

Solutions to preserve user's privacy in a CF system, propose to compute local aggregations of individual profiles for/by a community of users. The computed 'aggregate' is public and doesnt expose individual users profiles (e.g. Canny et al [7]). The aggregate is calculated iteratively by only adding vectors of user profile. Homomorphic encryption is used to allow sums of encrypted vectors to be computed and decrypted without exposing individual data. This work is however focused on a peer-to-peer framework, supposes that the majority of users are honest and states that the community should ideally know each other!

Another solution proposed by Polat et al [22] is to use randomized perturbation techniques to disguise the real user's consumption before sending them to the server. The server applies then a CF algorithm (SVD-based) on the disguised profiles and builds a matrix. To get a prediction for an item, the user sends a query to the server that computes a certain scalar product using the matrix and sends it back to the user. The latter computes the real prediction of the user likes/dislikes of the item using a particular formula involving the real private user profile. Authors indicate that the accuracy of their system is good but could be improved if more aggregate information about the user is disclosed along with the disguised data (i.e. knowing the real user preferences allows to generate more accurate recommendations).

Other approaches like that of Berkvosky et al [4] try to combine the obfuscation of a part of the user profile with the distribution of users profiles between multiple repositories to offer a CF system preserving user privacy while not hampering system accuracy. They studied what information of the user profile should not be hidden to continue having accurate recommendations. However, even though the generated recommendations were accurate, their results showed that a CF system working with real users' profiles (i.e. without any obfuscation) always provide better accuracy.

In summary, prior privacy-preserving collaborative filtering solutions either rely on heavyweight computations employing cryptographic operations or rely on degradation of recommendation quality by introducing perturbations in the users actual profile. In contrast, the collaborative-filtering approach of P3 is much more lightweight and works without requiring random perturbations to the user profile.

Collaborative personalized applications via anonymous peering with like-minded peers: In addition to collaborative-filtering recommendations, some Web2.0 applications like personalized web-search [19] could benefit from connecting the end-user with like-minded users with similar interests. The Gossple [5] system enables every end-user to be associated with a network of anonymous 'like-minded' acquaintances, and shows how applications like the personalized query-expansion can be built using Gossple. To achieve anonymization, Gossple uses a gossip-on-behalf approach where each node n is associated with a proxy P that gossips profile information on its behalf. The end-user's identity is hidden from P, by using an encrypted two-hop communication akin to onion-routing [24] wherein the end-user relays its profile information to P via an intermediate proxy that cannot decrypt the profile information. Although one could imagine using the Gossple substrate for designing a privacy preserving collaborative filtering application, the drawback of the Gossple's anonymous peering approach is that in order to know which two users are similar the complete set of consumed items (i.e. complete profile) consumed by the other user (although pseudonymized) is known to the other user. This is open to linkability attacks, wherein the pseudonym to user mapping can be inferred by sophisticated attacks that can intelligently infer the possible user based on the overall set of items consumed by him.

Scalable Collaborative Filtering: The Google-News personalization system [12] investigates how the scalability can be achieved in a recommender system that needs to deal with high item churn and large set of users. Their first technique to get scalability is the use of model-based algorithms for collaborative filtering wherein the user is mapped to one (or more) cluster(s) of 'like-minded' users using techniques like MinHash [9, 10] (an LSH [16] scheme for set-similarity using the Jaccard coefficient) and PLSI, and only the item-ratings of cluster members (instead of the entire set of users) is used to calculate the recommendation. Their second technique to get scalability is to develop MapReduce-friendly versions of the MinHash and the PLSI algorithms. In contrast to our P3 system, their system is a centralized system which assumes access to all the user's consumption history, and uses a MapReduce framework to scalably generate the recommendations. Although their PLSI algorithm cannot be trivially applied to our P3 setting because of the need of several global item consumption statistics, we observed that the MinHash algorithm can however be used to do local computation of cluster_ids as proposed in the realization of Step 2 of our P3 system.

5 Conclusion

We have proposed the design and realization of P3 system, a privacy-preserving-personalization middleware that enables the end-users to participate in a wide range of recommendation-based services, without privacy concerns of revelation of their sensitive profile information. We are currently in the process of evaluating P3's performance in terms of recommendation quality and overheads, and contrasting it with that of centralized recommender systems.

References

[1] A. Aghasaryan, S. Betge-Brezetz, C. Senot, and Y. Toms. A profiling engine for converged service delivery platforms. In *In Bell Labs Technical Journal (BLTJ), Volume 13, Issue 2, pages 93-103*, 2008.

[2] A. Aghasaryan, M. Kodialam, S. Mukherjee, Y. Toms, C. Senot, S. Betge-Brezetz, T. Lakshman, and L. Wang. Personalized application enablement by web session analysis and multisource user profiling. In *In Bell Labs Technical Journal (BLTJ), Volume 15, Issue 1, pages 67-76*, 2010.

[3] B. Applebaum, H. Ringberg, M. Freedman, M. Caesar, and J. Rexford. Collaborative, privacy-preserving data aggregation at scale. In *Proceedings of Privacy Enhancing Technologies Symposium (PETs)*, 2010.

[4] S. Berkovsky, Y. Eytani, T. Kuflik, and F. Ricci. Enhancing privacy and preserving accuracy of distributed collaborative filtering. In *Proceedings of ACM Conference on Recommender Systems (RecSys)*, 2007.

[5] M. Bertier, D. Frey, R. Guerraoui, A. Kermarrec, and V. Leroy. The gossple anonymous social network. In *Proceedings of ACM/IFIP/USENIX International Middleware Conference (Middleware)*, 2010.

[6] R. Caceres, L. Cox, H. Lim, A. Shamikov, and A. Varshavsky. Virtual individual servers as privacy -preserving proxies for mobile devices. In *Proceedings of ACM SIGCOMM Workshop on Networking, Systems, Applications on Mobile Handhelds (Mobiheld)*, 2009.

[7] J. Canny. Collaborative filtering with privacy. In *Proceedings of IEEE Symposium on Security and Privacy (SP)*, 2002.

[8] M. Charikar. Similarity estimation techniques from rounding algorithms. In *Proceedings of ACM Symposium on Theory of Computing (STOC)*, 2002.

[9] E. Cohen. Size-estimation framework with applications to transitive closure and reachability. In *Journal of Computer and System Sciences 55*, 1997.

[10] E. Cohen, M. Datar, S. Fujiwara, A. Gionis, P. Indyk, R. Motwani, J. Ullman, and C. Yang. Finding interesting associations without support pruning. In *Proceedings of International Conference on Data Engineering (ICDE)*, 2000.

[11] F. Dabek, B. Zhao, P. Druschel, J. Kubiatowicz, and I. Stoica. Towards a common api for structured peer-to-peer overlays. In *Proceedings of International Workshop on Peer-to-Peer Systems (IPTPS)*, 2003.

[12] A. Das, M. Datar, A. Garg, and S. Rajaram. Google news personalization: Scalable online collaborative filtering. In *Proceedings of International World Wide Web Conference (WWW)*, 2007.

[13] R. Dingledine, N. Mathewson, and P. Syverson. TOR: The second generation onion router. also www.torproject.org. In *Proceedings of USENIX Security Symposium (Usenix Security)*, 2004.

[14] Foursquare. http://www.foursquare.com.

[15] M. Fredrikson and B. Livshits. RePriv: Reimagining content personalization and in-browser privacy. In *Proceedings of IEEE Symposium on Security and Privacy (SP)*, 2011.

[16] A. Gionis, P. Indyk, and R. Motwani. Similarity search in high dimensions via hashing. In *Proceedings of International Conference on Very Large Data Bases (VLDB)*, 1999.

[17] S. Guha, B. Cheng, and P. Francis. Privad: Practical privacy in online advertising. In *Proceedings of USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2011.

[18] iGoogle. http://www.google.com/ig.

[19] A. Mislove, K. Gummadi, and P. Druschel. Exploiting social networks for internet search. In *Proceedings of Workshop on Hot Topics in Networks (HotNets)*, 2006.

[20] A. Mislove, G. Oberoi, A. Post, C. Reis, P. Druschel, and D. Wallach. AP3: Cooperative, decentralized anonymous communication. In *Proceedings of ACM SIGOPS European Workshop (SIGOPS)*, 2004.

[21] Netflix. http://www.netflix.com.

[22] H. Polat and W. Du. Svd-based collaborative filtering with privacy. In *Proceedings of ACM Symposium on Applied Computing (SAC)*, 2005.

[23] K. Puttaswamy, R. Bhagwan, and V. Padmanabhan. Anonygator: Privacy and integrity preserving data aggregation. In *Proceedings of ACM/IFIP/USENIX International Middleware Conference (Middleware)*, 2010.

[24] M. Reed, P. Syverson, and D. Goldschlag. Anonymous connections and onion routing. In *IEEE Journal on Selected Areas in Communications (JSAC)*, 1998.

[25] Stumbleupon. http://www.stumbleupon.com/.

[26] V. Toubiana, A. Narayanan, D. Boneh, H. Nissenbaum, and S. Barocas. Adnostic: Privacy preserving targeted advertising. In *Proceedings of Network and Distributed Systems Security Symposium (NDSS)*, 2010.