CNS: A Content-centric Notification Service

Jiachen Chen*, Mayutan Arumaithurai*, Xiaoming Fu*, K.K.Ramakrishnan†
*University of Goettingen, Germany, †AT&T Labs-Research, U.S.A.

I. Introduction

Pub/sub systems have become popular. Supporting such applications can be a significant challenge because of the need for timeliness and the need for dynamic user relationships. Server based pub/sub solutions such as Twitter pose scalability challenges. They place a large load on the server, including the need for the server to support frequent polling by all the subscribers. Overlay solutions (P2P or overlay multicast) are topology unaware and therefore might introduce overhead in the network as well as increase latency. Having a network infrastructure provide the support for these dynamic and large scale pub/sub systems is therefore desirable.

Content Centric Networking (CCN) [1] is a novel networking paradigm centered around content distribution rather than host-to-host connectivity. This change from a *host-centric* to a *content-centric* communication capability removes the need for receivers to know and establish context with specific sources of information. Named Data Networking (NDN) [2] is one of the popular CCN approaches.

Our recent proposal of a Content-Oriented Publish/Subscribe System (COPSS) [3] enhances some of the existing CCN approaches such as NDN with a push-based multicast capability. COPSS uses the notion of hierarchical Content Descriptors (CDs) to name the components of information disseminated. CDs are utilized both by users to subscribe and for users to publish information. Publishers need not have apriori knowledge of the intended recipients. COPSS is designed to facilitate a highly dynamic and large scale pub/sub environment and is able to deliver content in a timely manner.

In this paper we develop a Content-centric Notification Service (CNS) that can be used for timely notification of essential information from a user to a group of subscribers that have requested (or potentially will request) that information. We envisage such capabilities to be desirable in a variety of circumstances, including disaster management, event notifications and many other scenarios where timely notification is key. Effective authorization is also a very important aspect in such an environment, *i.e.* certain publishers would prefer having control over who has visibility to their information. *E.g.*, a publisher might want to publish personal data only to close friends, or another set of data to office colleagues and yet another set to all his acquaintances. In some special cases, especially in cases where energy efficiency is critical (*e.g.*, a user device's battery has little residual power), the publisher might not want to (or be able to) do the authorization himself. The system should also allow a 3^{rd} party (either the network or a friend) to perform the authorization function. We explore enhancing our COPSS pub/sub environment with a preliminary authorization framework that could be used as the building block for a full-fledged notification service that could be used in future disaster management situations.

In this work, we demonstrate how CNS facilitates pub/sub based notification capabilities, highlighting the enhancement of the basic CCN functionality to achieve *control efficiency* achieved with the use of hierarchical CDs. *Network efficiency* is achieved with the use of mutlicast, and *timeliness* is achieved with the push-based multicast approach. Furthermore, we will demonstrate the basic authorization capability to provide access control for a publisher's information.

II. CNS: A Content-centric Notification Service

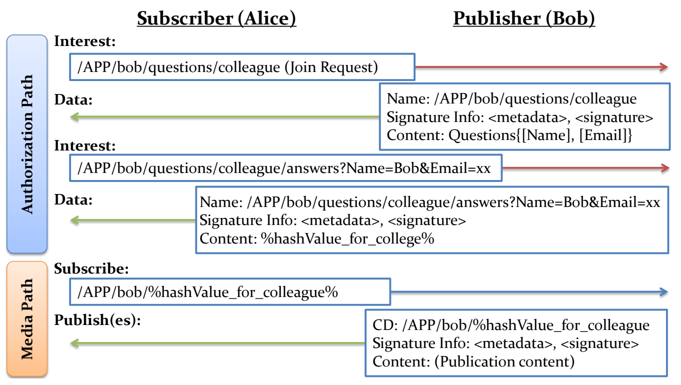
CNS leverages the COPSS architecture for efficient data dissemination. Here, we describe the modules of CNS.

Publisher Management: Every user in our application has a unique user (publisher) ID so that with a simple combination of `/APP/%UserID%`, we can get a unique prefix for that user. This can be part of the Content Name that is used by both publishers and subscribers. The Content Name may have other parameters such as the publisher name, signature etc., as specified in NDN [2]. Moreover, it may also include CDs, information that goes beyond what is in the data, so that you can identify the publisher and who has signed it; it may also include a hash to ensure data integrity; it may also have some additional information that indicates the version or sequence number etc. The published data (*e.g.*, a document) itself comprises many tokens (*e.g.*, keywords), some of which are used as CDs by the publisher.

A user will have a **profile** which is a set of key-value pairs of his personal properties to uniquely identify the user. The user may create several **groups** according to his or her preference. For every group, the user may choose a combination of the properties that identify the user, as the authorization token. Only the users who know the values of the authorization token associated with the group can subscribe to that group.

E.g., user Bob has the prefix of `/APP/bob`. His profile contains properties of Email, cell phone number, favorite book, favorite football team, *etc.*. Using a subset of these, he can have authorization tokens for groups that are his friends, colleagues, families, *etc.*. For the 'colleague' group, Bob may use his name and Email as the authorization token. But for people who want to subscribe to his 'friend' group, they need to know his name and cell phone number as the authorization token.

Basic Authorization: To prevent malicious users from subscribing to a (well-known) CD, our CD based group name is renamed with the use of hashes (on the group name + user ID) to provide a basic access control wherein only subscribers authorized by the publisher gain access to this CD. As shown in Fig. 1, when a subscriber wants to subscribe, he sends a request to the publisher with the group name (`/APP/bob/questions/colleague` in the example), and the publisher responds with a challenge, *e.g.* "what is my `Name` and `EmailId`?", where "Name" and "EmailId" are the two attributes in the profile the publisher selects as the authorization token used to authorize the subscribers. Assuming that the subscriber knows this information (which means he is eligible to subscribe to that group), he responds with the appropriate values in the form of HTML parameters: `answers?Name=Bob&EmailId=bob@icn.org`. The publisher on verifying this data sends him the hash-based hierarchical CD, which the subscriber then subscribes to. If the subscriber cannot answer the questions correctly, the publisher will respond with an error instead.



Subscriber (Alice)
Publisher (Bob)
Authorization Path
Interest:
/APP/bob/questions/colleague (Join Request)
Data:
Name: /APP/bob/questions/colleague
Signature Info: <metadata>, <signature>
Content: Questions{[Name], [Email]}
Interest:
/APP/bob/questions/colleague/answers?Name=Bob&Email=xx
Data:
Name: /APP/bob/questions/colleague/answers?Name=Bob&Email=xx
Signature Info: <metadata>, <signature>
Content: %hashValue_for_college%
Media Path
Subscribe:
/APP/bob/%hashValue_for_colleague%
Publish(es):
CD: /APP/bob/%hashValue_for_colleague
Signature Info: <metadata>, <signature>
Content: (Publication content)

Fig. 1: Protocol exchange for Alice follows Bob's Colleague group

Note that the authorization path uses the query/ response approach of NDN. The network provides the functionality of caching on both challenge and the group hash value. On receiving a correct answer (in the form of an Interest with the same name as a previous correct answer), a router in NDN can respond directly if an existing copy of the hash value is already in the Content Store. This can save substantial processing and communication overhead on the publisher end (imagine the case a pop star that has to authorize a million fans who want to subscribe to his 'fan' group, possibly with a cell phone!).

Data Dissemination & Offline Support: Our CNS demo leverages COPSS to provide a large scale timely notification service. Along with the content published, the user includes the authorization token, which looks like a CD (*e.g.* `/APP/bob/%hashValue_for_colleague%`). Bob can publish to all his colleagues using a Publish packet with this authorization token. As defined in COPSS, Bob can send packets to multiple groups by putting all the needed authorization tokens in the same Publish packet. The network will ensure the dissemination of the information automatically.

To help new subscribers as well as users that come online after having been disconnected for a period of time, we leverage the 'broker' concept suggested in COPSS. The broker subscribes to all the messages published by the user (*i.e.*, CD `/APP`) and stores them for a period of time. The new subscribers can query the broker on completing the authorization phase. The name of the Interest packet is in the form of:

`/Broker/APP/UserID/GroupHash/MessageRange.`

A subscriber can also get online periodically and request missing messages by polling the broker. This can help save energy on the subscriber's device (*e.g.*, battery) in extreme cases.

3^{rd}-Party Authorization: In some scenarios, a publisher might want to delegate the task of authorization to a third party (a subscriber in the same group in our demo). When a publisher is not able to respond to authorization requests, a subscriber (S_{req}) could forward the request to another subscriber (S_{del}) that is in the same group. Since S_{del} knows the answer for joining the required group, he can decide if the answer provided by this new subscriber is correct to then provide the hash of the authorization token to S_{req}.

However, there can be security concerns associated with such transitive authorization schemes. Publishers might not want to delegate the authorization on some groups, nor do they want to delegate the authorization to some subscribers. Since our demo targets as a building block for a generic notification service, we have not yet addressed these issues in providing a

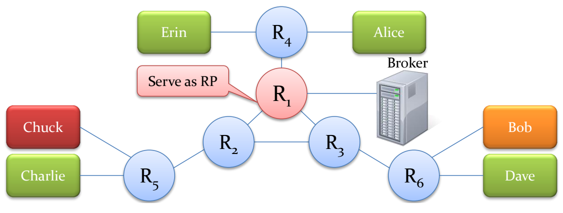


Fig. 2: Demonstration Topology

3^{rd} party authorization mechanisms, using approaches like a reputation systems. Instead, we use a simple solution currently: a publisher can specify if a group is allowed to provide transitive authorization; and the 3^{rd} party authorization is done manually (*i.e.* S_{del} decides if S_{req} can join a group by clicking a button in our demo).

III. Demonstration

We will show with a live demonstration the benefits of COPSS based content-centric notification service with authorization. Our topology (see Fig. 2) consists of 6 COPSS enabled routers (in which 1 serves as an RP); 6 clients, 1 of which 1 is named "Bob" will act as a publisher (by design, any client could be a publisher or a subscriber, but for the purpose of the demo, we will choose Bob to be the publisher). Alice and Charlie are Bob's close friends, Dave is his colleague. We will show how subscribers gain access to their respective groups and how Bob is able to post messages using the COPSS environment and how the subscribers are able to receive messages posted to group they are authorized to be part of. The benefit of using COPSS is in terms of control efficiency and message delivery efficiency as well as timeliness, which our demo will highlight. All users in our demo can be offline and acquire the missing messages when they are back online. We will also seek to show a situation where Erin will try to obtain an authorization when Bob is not online, through the 3^{rd} party authorization mechanism. Chuck is a malicious node that will also attempt to gain access to the messages posted by Bob, but fails in the authorization phase.

Furthermore, in order to present the work at a larger scale, we leverage a trace-driven demonstration with roughly 20 clients. In this demonstration, every user is both a publisher and a subscriber. Due to the scale, we can show a more complicated (and dynamic) subscription status to highlight the benefits of a "content-centric" pub/sub notification service.

IV. Conclusion

This work demonstrates the benefit brought by COPSS for an efficient notification service, including the convenience of hierarchical group management, network efficiency and timeliness in delivering the notification. The work will also demonstrate a simple first-step authorization. Future work will include a more complete authorization, authentication, encryption and mobility support for a scalable notification service.

Acknowledgment

The work for this paper was performed in the context of the FP7/NICT EU-JAPAN GreenICN project.

References

[1] V. Jacobson, D. K. Smetters *et al.*, "Networking Named Content," in *CoNEXT*, 2009.

[2] L. Zhang, D. Estrin *et al.*, "Named Data Networking (NDN) Project," PARC, Tech. Report NDN-0001, 2010.

[3] J. Chen, M. Arumaithurai *et al.*, "COPSS: An Efficient Content Oriented Pub/Sub System," in *ANCS*, 2011.