Gaming with COPS: A Content Centric Communication Infrastructure for Gaming Applications

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

I. Introduction

Massively Multiplayer Online Role Playing Games (MMORPG) [1] are quite popular due to the their well structured and creative scenarios and the realization of real-world human interactions such as communities, war, family, ally, merchandizing. World of Warcraft and Counter-Strike are examples of such games and are characterized by the need for a short latency since every action a player performs needs to be broadcasted to all the other players that are viewing the same sub-world. These games involve a large number of players and require a persistent world that is usually hosted by the game's publisher thereby incurring a heavy load on the servers for player management and data transfer. Deploying such a game in a decentralized/P2P environment is not a straight forward task since it is difficult to obtain knowledge about the other players in the same sub-world and to efficiently disseminate the information to them.

Content Centric Networking (NDN) [1], [2] is a novel networking paradigm centered around content distribution rather than host-to-host connectivity. This change from *host centric* to *content centric* decouples the action receivers from the performers. In this work, we attempt to build a content centric communication infrastructure for a decentralized gaming environment wherein the players can directly push the content to the other players in the same sub-world in an efficient manner. The role of the server is reduced to that of maintaining an update of the world and providing the snapshot to the players who enter a sub-world. To achieve this, we make use of a content Content-Oriented Publish/Subscribe System (COPS) [3], which enhances NDN [2] with push-based dissemination, as the communication layer of MMORGs.

II. COPS: A Content Centric Communication Infrastructure

We first give a brief introduction to NDN and COPS.

A. Content-Centric Networking

NDN uses hierarchical human-readable *ContentName*s to address content items, e.g., `/conf/papers/COPS.pdf`. There are two NDN packet types, *Interest* and *Data*. A consumer queries for content by sending an Interest packet and a provider responds with a Data packet and the Data packet consumes this Interest packet. Data 'satisfies' an Interest if the ContentName in the Interest packet is a prefix of the ContentName in the Data packet. Consumer is decoupled from providers since they only ask for the content rather than query from a specific host.

NDN requires a new forwarding engine to perform the basic operations. The forwarding engine contains FIB (Forwarding Information Base), Content Store and PIT (Pending Interest Table). FIB is used to forward Interest packets toward potential source(s) of matching Data. Content Store is the same as the buffer memory of an IP router but tries to remember the Data packets as long as possible, which works like a cache in the network. PIT keeps track of 'bread crumbs' of Interest so that Data packets follow to reach back the original requester(s).

B. Content-Oriented Publish/Subscribe System

To achieve more efficiency in transmission, COPS enhances NDN with push-based dissemination. This also relieves the consumers in NDN from knowing the name of every piece of data beforehand. Instead, they express interests to *Content Descriptor*s (CDs), e.g., `/sports/soccer`. Data providers (publishers) send announcements to a CD when they have a new piece of data. The CDs are grouped in hierarchical structure so that subscribers of higher level CDs can also receive announcements of lower level CDs, e.g., a subscriber of `/sports` can also receive announcements of `/sports/soccer`, `/sports/swimming`, etc.

COPS aware routers are equipped with a subscription table (ST) that maintains CD-based subscription information downstream of them in a distributed, aggregated manner, as in IP multicast. And COPS was proved to be more efficient than IP multicast because of the hierarchical CD management.

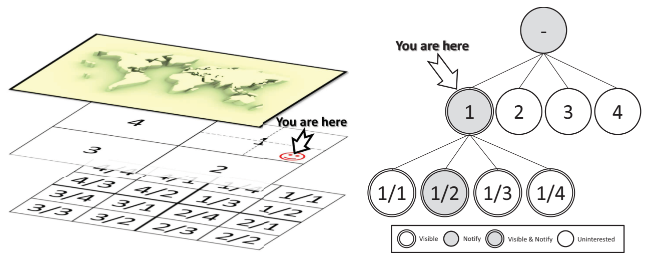
COPS also provides subscriber offline support which allows 'asynchronous' data dissemination. It enables the user to receive messages that were missed while being offline. COPS supports this by having a dedicated broker that acts as a store for all COPS multicast messages.

III. GoCOPS: A P2P Game over COPS

A. GoCOPS Overview

The basic assumption we make is that all the players share a same map downloaded beforehand. For practical reasons such as efficient broadcast of updates, the world-map is divided into

[1]http://en.wikipedia.org/wiki/Massively_multiplayer_online_role-playing_game



(a) Map division. (b) Logical hierarchy.

Fig. 1: Sub-world hierarchy.

various zones and each of these zones can be divided further into areas and smaller areas thereby forming a hierarchy. We use the term sub-world to represent these smaller zones and ares. Such kind of a mechanism is used to ensure that the players receive update pertaining to the sub-world they belong to, e.g., a player in a room gets to see all the other players in the same room, whereas a player flying over the room gets to see those players as well as others in the vicinity. Furthermore, the sub-world is composed of objects that form the smallest unit. This ensures that only the changes on the objects need to be broadcasted instead of the whole sub-world.

Fig. 1a shows a world map which is first divided into 4 zones (marked $1-4$) and each zone is further divided into 4 areas (marked $1/1-4/4$). Fig. 1b shows the logical hierarchy (areas belonging to zone 2, 3 and 4 are omitted). When a player is flying over area $1/2$ on the zone layer, he can see players in $1/1-1/4$ (standing on the area layer or flying on the zone layer). In Fig. 1b, the visible sub-worlds are in double-circle. When he modifies some object in $1/2$, the update will be multicasted to all the players standing on $1/2$, flying in 1, and on the top. The gray circles are the notified sub-worlds in Fig. 1b.

Below, we describe how to use COPS in gaming.

B. Sub-world = Hierarchical CD

In GoCOPS, sub-world is managed in a hierarchical structure. So it is easy to map sub-world to hierarchical CDs directly. A player can be publisher when he performs some action while at the same time a subscriber since he is in some sub-world and needs to receive updates. The one-step communication model in COPS is used to disseminate such actions.

E.g., if a player flying over area $1/2$ at zone layer, he will subscribe to CD `/map/1`. According to the hierarchical definition of COPS, any message belonging to $1/1-1/4$ will be disseminated to him. But when he modifies some object in area $1/3$, the action will be multicasted to CD `/map/1/3`, so that the subscribers of CD `/map`, `/map/1` and `/map/1/3` will receive the action. This conforms to the definition of 'share a sub-world'. Note that the player only sends one packet and all three groups will receive it. But in IP multicast the player will have to send out three different packets and in a server-based communication the player will have to send a packet to every player that can see the action.

C. Conflict Elimination & Snapshot Managing $\rightarrow$ Broker

However, there are differences in requirements between gaming and publish/subscribe system. We modify the broker for user offline described in [3] to solve two major problems in gaming: snapshot managing and conflict resolution.

Snapshot Managing: It is natural for a player to move from one sub-world to another. At the time he enters (or approaches) a new sub-world, he should be able to see the current status of the sub-world, which we call a snapshot. Here, we use the broker to manage the snapshot of the sub-world. When a player performs an action, he sends a packet to the broker instead of doing the multicast himself. The broker then modifies its snapshot and multicasts the action. When a player enters the sub-world, he will query for the latest version of the snapshot from the broker. Caching in NDN can reduce the load on broker and network if multiple players wish to download a same version of the snapshot.

Conflict Resolution: In online gaming, players tend to make conflict actions like collision or modify a same object at the same time. A common practice to resolve such kind of conflicts is to discard the action arrives server later. In our solution, we use the broker to check for conflicts when it is modifying the snapshot. When a player performs an action, he will send the action to the broker but not change his local view. On receiving the action, the broker checks for conflicts and multicasts the action (we call it a result) if no conflict exists. The action performer will modify the local view only when he receives the result resulting in the natural discarding of conflicts.

GoCOPS Broker $\neq$ Server: The functionality of GoCOPS brokers is different from that of the normal game servers mainly because of the following 2 reasons:

- Brokers do not manage players in the sub-world. The player (his gaming client) is responsible for managing the CDs he should subscribe to and the CDs he should multicast to. Brokers only manage the snapshot of the sub-worlds (and at the same time check for conflicts). This relieves the brokers from the burden of player management and event dissemination, thereby reducing the computation cost and network traffic.
- Due to the hierarchical map formation, brokers can be decentralized and off-loaded naturally. As the load on the broker increases, it can offload some CDs (sub-worlds) to a new broker. Since the architecture is built on COPS and players send $Interest$s to the ContentName of a CD (rather than broker address); they will not be affected by the introduction of new brokers. The new broker only need to express that he serves the new CDs. The COPS network will accordingly modify the FIB and ST. The packets of the users will be redirected automatically to the new broker and a new multicast tree can be formed at the 1^{st} hop router of the broker.

References

[1] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking named content," in *CoNEXT*, 2009.

[2] L. Zhang, D. Estrin, J. Burke, V. Jacobson, and J. Thornton, "Named data networking (ndn) project," PARC, Tech. Report NDN-0001, 2010.

[3] J. Chen, M. Arumaithurai, L. Jiao, X. Fu, and K. K. Ramakrishnan, "Cops: An efficient content oriented publish/subscribe system," University of Goettingen, Tech. Rep., 2011.