# Matryoshka: Design of NDN Multiplayer Online Game

Zhehao Wang
REMAP, University of California, Los Angeles
102 East Melnitz Hall
Los Angeles, CA 90095
zhehao.mail@gmail.com

Zening Qu
REMAP, University of California, Los Angeles
102 East Melnitz Hall
Los Angeles, CA 90095
quzening@gmail.com

Jeff Burke
REMAP, University of California, Los Angeles
102 East Melnitz Hall
Los Angeles, CA 90095
jburke@remap.ucla.edu

## ABSTRACT

Massive multiplayer online games (MOG) have become increasingly popular over the past decade. Peer-to-peer structures were explored for commercial online games. However, maintaining security and availability while scaling users has driven most multiplayer online games towards a client-server or client-superpeer architecture.

Client-server multiplayer games face certain problems: a small number of points of failure and traffic centralizing at several servers. Users of popular games complain about the decrease in quality of service, largely caused by these two factors. In order to tackle the problems of traditional client-server online games, this demo presents Matryoshka, a pure peer-to-peer multiplayer online game using the named data networking[1] (NDN) future internet architecture.

NDN has several major strengths over IP; among them are natural multicast support, content-based security and mobility support. By utilizing the strength of multicast and content caching, we believe that a pure peer-to-peer MOG design in NDN can avoid challenges and limitations found in IP.

Synchronization in a serverless distributed environment is a key problem for pure peer-to-peer structure. Namespace synchronization in just such a situation has been studied for other serverless NDN applications like chat and file sharing. The ChronoSync[2] model is proposed for both use cases. Other cases like vehicular network also study synchronization in a physical environment.

The challenges faced by an online game are different, which we explore in this project. In this case, the environment is a virtual world. Each player has an area of interest, and it only needs to know things in this virtual area instead of everything happening in the game, and this we define as ‘locality in the game world’. In addition, players whose areas of interest intersect with each other should reach consistent conclusions about things in the intersected area, which introduces the synchronization problem.

NDN’s content caching and natural multicast support feature may facilitate the distribution of game synchronization data. We utilize these features by statically and recursively partitioning the whole virtual environment into octants, thus providing a shared namespace for every peer running the game. Then, all the peers that care about the same region can share the data brought by synchronization interests towards the same nodes in the octree. Figure 1 presents the octree partition of the game world.

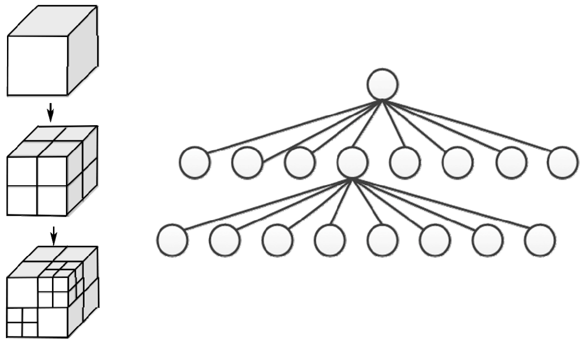
**Figure 1 Octree partition of the game world**

Then, we apply a two-step synchronization to deal with the two questions which each peer addresses the network: “which players are in my vicinity” (*discovery*) and “what are those players doing” (*update*). Below we explain this mechanism and present the namespace design.

For the first question, peers who care about the same octant synchronize their name dataset belonging to the octant. To do this, discovery interests containing the octant indices and a digest of the octant’s set of object names are expressed periodically to all peers in a “broadcast” namespace. Peers receiving the discovery interest respond with their own set of object names, if they have different digests for the octant.

The namespace for discovery is given in Figure 2. The top-level game name component separates the game into several sub-worlds, and only players in the same sub-world need to discover each other. Below the sub-world are

*ICN’14*, September 24–26, 2014, Paris, France.
Copyright © 2014 ACM 978-1-4503-3206-4/14/09…$15.00.
http://dx.doi.org/10.1145/2660129.2660139

octant indices, which indicate the octant's absolute location in the game world. For each interest, we append a digest component, which contains the hash of the set of object name strings in that octant. Every peer should have the same hash for octants belonging to their intersection, when steady state is reached.

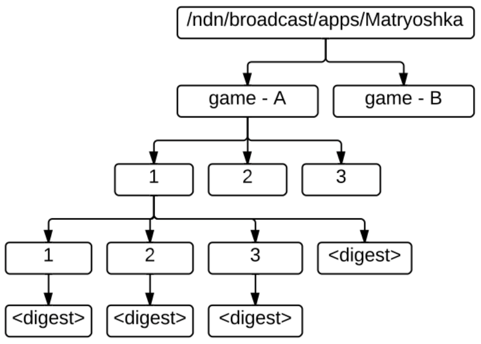

**Figure 2. Broadcast discovery namespace**

Using the object names returned in response to the discovery interest, update interests are expressed by each peer on an ongoing basis for the virtual location of the players and several non-player characters (NPC) that remain in their area of interest. The requesting peer decides whether the resulting objects should be recorded and rendered in the local game instance.

Figure 3 demonstrates the update namespace. Each physical peer running the game is represented by a globally unique process name. In each process, a variety of objects—e.g., player and NPCs as well as other elements of the game—are hosted. For each object, interests are expressed for its position and actions using the namespace as shown. Position and action names follow NDN versioning conventions, enabling interest selectors to be used to ensure the latest version is received by the requesting peer.

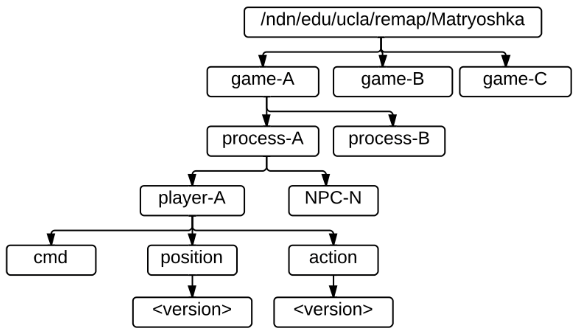

**Figure 3. Position and action update namespace**

Our approach also explores techniques for progressive discovery, an optimization for leveraging the usage of larger octants, as well as dynamic adjustment of the area of interest for a player.

Representing a player's spherical areas of interest with only leaf octants can cause the amount of discovery interests to be large, thus increasing synchronization traffic. Progressive discovery aims to allow peers to issue interest packets corresponding to larger octants, so that we can approximate spherical areas of interest with fewer interests. In this case, a peer may receive synchronization interests for large octants, about which it may have incomplete knowledge. Our approach balances the impact of unanswered interests vs. incomplete responses by having answering peers wait to reply with a delay proportional to the completeness of their knowledge of the requested octant. Further, the requesting peer adjusts the area of interest based on response performance. For example, when the latency in discovery interests getting answered is large (suggesting few peers with knowledge), or the number of objects in a player's area of interest exceeds a preconfigured threshold (resulting in a lot of traffic), the game application automatically shrinks the area of interest.

The demo application *Matryoshka*, a game environment implementing the design outlined above, was built using Unity3D game engine, and ndn-dot-net, a C# adaptation of NDN Common Client Library. The demo will show the game code running on a small number of peers, and with a visualization of the network traffic going on in the two namespaces as players navigate the game world on each peer.

For the demo, player characters and NPCs are instantiated by each peer, and each peer can navigate around the common world using their player character. Player and NPC discovery, and position update under several preferences and scenarios will be demonstrated. Challenging scenarios to handle with static octree partitioning, such as having players that are close to the border between two sub-regions of the highest subdivision hierarchy, will be shown in addition to easier to handle situations.

## Categories and Subject Descriptors

C.2.4 [**Distributed Systems**]: Distributed Application

## Keywords

Massive multiplayer online game; named data networking; synchronization

## REFERENCES

[1] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, R. L. Braynard. 2009. Networking Named Content. CoNEXT 2009, Rome, Dec. 2009. DOI=http://doi.acm.org/10.1145/1658939.1658941.

[2] Z. Zhu, A. Afanasyev. 2013. Let's ChronoSync: Decentralized dataset state synchronization in Named Data Networking. ICNP 2013, Oct. 2013. DOI=http://dx.doi.org/10.1109/ICNP.2013.6733578.