# Multiplayer Online Games Over Named Data Network

Zening Qu
quzening@remap.ucla.edu

Jeff Burke
jburke@remap.ucla.edu

❊ Maintain game consistency among players. ❊ Exploit emerging Sync primitives.
❊ Inform NDN-based game development and synchronization protocol choices.

## Background

Multiplayer online games (MOGs) are interesting as an application domain of Named Data Network (NDN):

- Huge amount of packets need to exchanged at high frequency by each instance of MOG. This makes games an appropriate test for NDN distribution efficiency.
- Synchronization mechanisms are needed for MOGs to maintain game state consistency, and they exhibit fundamental influences on the game performances. Emerging NDN primitives such as Sync[1] can be exploited and evaluated in the game context.
- MOGs are also good test cases for the NDN security model. Player authentication and cheating prevention are of special importance to MOGs.

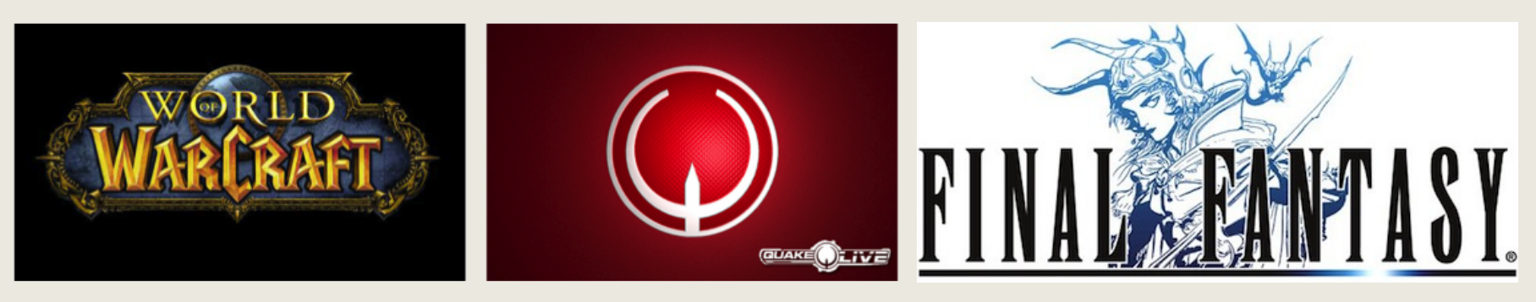

**Figure 1:** Examples of multiplayer online games: World of Warcraft[2], Quake[3]and Final Fantasy[4].

## Past Work

This project explored ways to easily synchronize a simple multi-player car racing game in a peer-to-peer fashion.

- CCNx Sync was employed for game discovery as well as object discovery.
- Basic Interest/Data exchange was used for state synchronization (see section Synchronization).

Project products include:

- Egal Car, a prototype car racing game (see figure 2);
- Egal, C# bindings for Xerox PARC's CCNx library, including its new Sync protocol, usable in Unity game engine.

**Figure 2:** Screenshots of Egal Car, the project's prototype game.

## Future Work

Goals for future work:

- Explore the synchronization mechanism of more complex game events such as inter-user and user-object interactions.
- Evaluate the scalability of NDN-based peer-to-peer games.
- Study authentication and anti-cheating problems in more detail and provide some preliminary solutions based on NDN's signature-based trust model.

These goals lead to the design of a new prototype game:

- Game genres such as role playing games (RPG) are preferred, as they usually contain rich interactions between users and objects.
- A capacious scene is needed to test the game's scalability (the ability to hold a large number of concurrent players).
- Reasonably complex gameplay is needed to hint and simulate the realistic security and synchronization requirements of MOGs.

Please see section New Prototype for the design of our new RPG game.

## Synchronization

Two classes of data updates shall be synchronized in Egal Car:

- *Asset*s, which change slowly and unpredictably;
- *State*s, which change quickly and predictably;

Their names form a tree in the game's namespace (see figure 3).

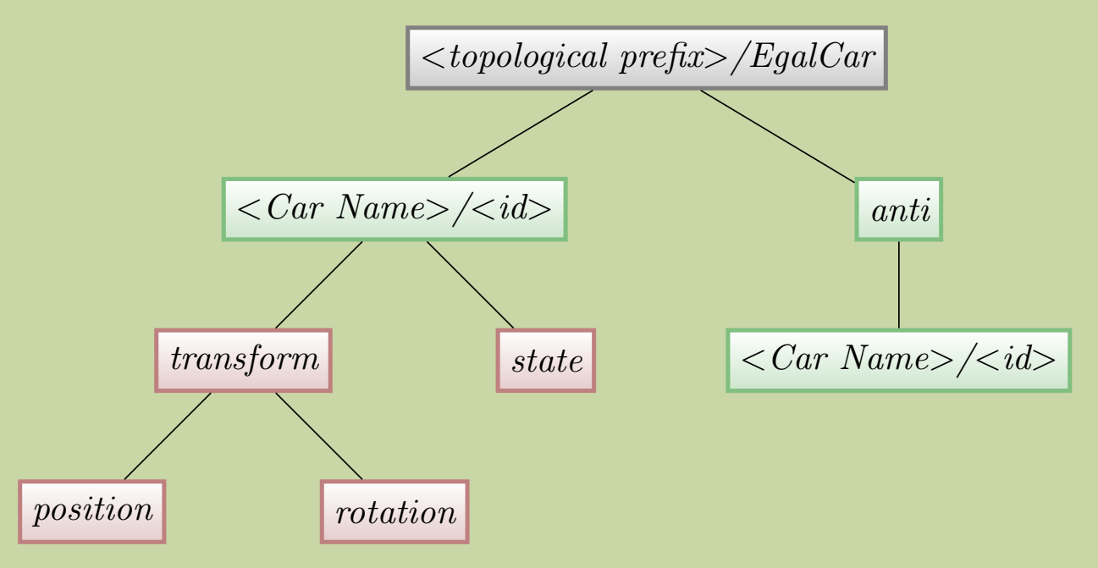

**Figure 3:** Part of namespace of Egal Car. Asset names and State names are shown in green and red respectively.

| Game Functionality | Sync Functionality |
|---|---|
| Game Discovery | Broadcast Sync root advise Interests |
| User Join | Add a new object to the Sync slice |
| User Quit | Add an anti-object to the Sync slice |

**Table 1:** Egal Car game functionalities and the corresponding Sync actions.

| Synchronization Mechanism | Ordered? | Reliable? |
|---|---|---|
| Asset Synchronization | × | ✓ |
| State Synchronization | ✓ | × |
| Event Synchronization | ✓ | ✓ |

**Table 2:** Comparing Asset, State and Event synchronization.

### Asset Synchronization

Asset synchronization, specifically player entry and exit, is implemented with Sync in a broadcast namespace (see Table 1).

This mechanism is unordered yet reliable, as Asset updates are *uncorrelated* in Egal Car.

### State Synchronization

State updates are snapshots of asset/global properties which are also uncorrelated. State synchronization should be ordered for visual rendering consistency but does not need to be reliable as newer updates will always *overwrite* the older ones.

Normal NDN Interest/Data packets are used for implementation:

- *Exclusion Filter:* exclude updates that are already learnt
- *Rightmost Child Selector:* request the latest update, with undelivered ones dropped

### Event Synchronization

Event synchronization the future model to describe updates that are *correlated*—user-user interactions and user-object interactions.

This mechanism shall be ordered and reliable. In addition to timestamps, sequence number will be incorporated into names for peers to track event update series. Rollbacks will be performed for event updates that arrive late.

Please see Table 2 for a comparison of the three mechanisms.

## Architecture

Egal Car is adapted from Unity's single-player car race tutorial[5]. The gameplay has been left intact, and the project's network module makes the tutorial a Peer-to-Peer multiplayer online game.

The network module builds on the CCNx libraries (see figure 4):

- *CCNx Daemon* provides basic Named Data Networking support, such as sending and receiving Interest and Data packets;
- *Repository Daemon* manages read/write requests;
- *CCNx Sync* manages repository contents.

The Egal network module is where asset and state synchronization is implemented. It contains C# scripts for Unity and two C libraries: CCNx Library and Egal Library. The InteropServices of the .NET Framework are used to invoke C functions from C# and vice versa.

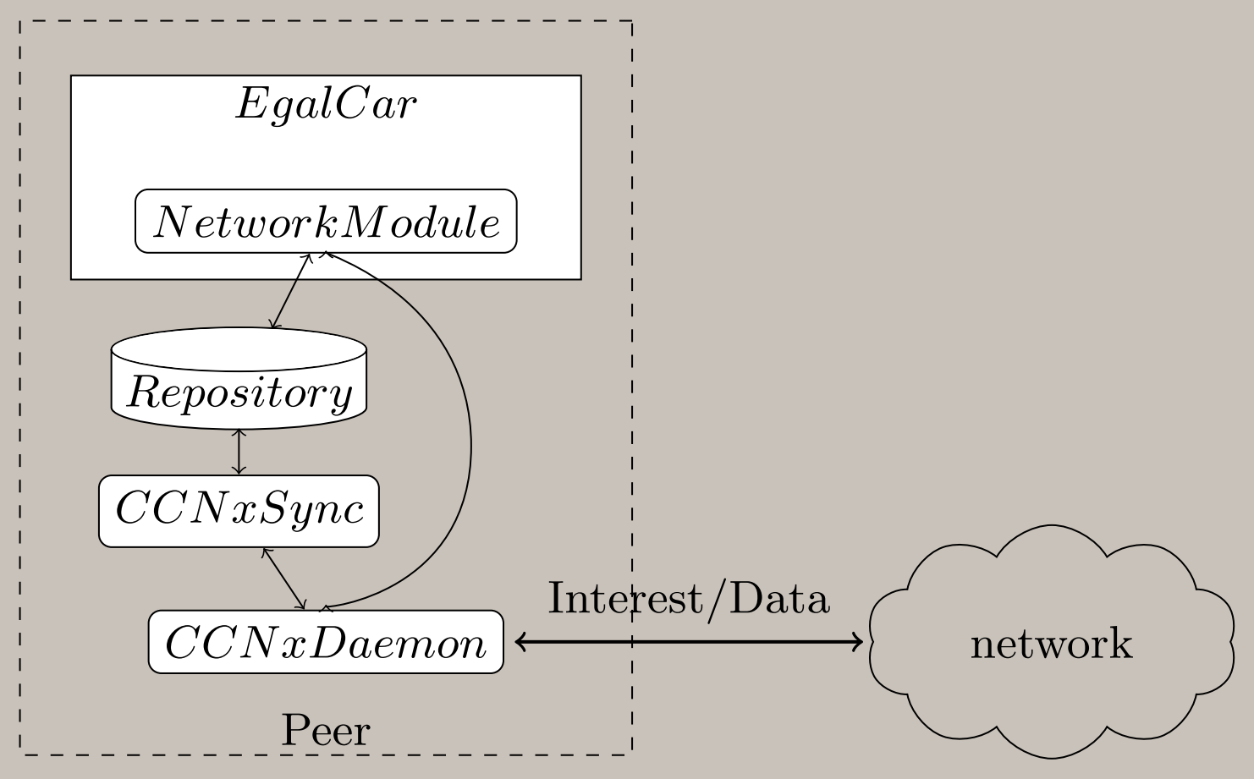

**Figure 4:** Architecture of prototype game application.

## New Prototype

The new game prototype is an RPG in which gameplay is inspired by Le Petit Prince (see figure 5).

Real-world data will be used to simulate the large-scale dynamic creation of game objects and the real-time decisions made by players.

**Figure 5:** Concept sketches of the next prototype game.

## Conclusions

- The workflow of NDN-based multiplayer game development was explored, including namespace design and name classification.
- Unordered collection synchronization approaches hold promise for streamlining development and performance of multiplayer online games and distributed simulation, especially in simplifying development and deployment of peer-to-peer architecture.
- The impacts of the unreliable nature of basic interest/data exchange were explored, which shows its strengths in synchronizing uncorrelated, quickly changing data.
- The mechanisms were implemented in a prototype game, Egal Car, and organized into an open-source synchronization library, Egal, to support future NDN game developers.
- A new model is required for the synchronization of correlated data updates. The proposed model is Event synchronization, which trades off interactivity for consistency.
- A more extensive game prototype planned to test the new synchronization model and study the related problems such as authentication and anti-cheating.

## Acknowledgements

Thanks to Alex Horn and Alessandro Marianantoni of REMAP for their help testing the sample application, Ren Rong for the sketches of the next prototype game, and Professor Lixia Zhang for feedback on the architecture. We are grateful to NSF for generous funding under NSF FIA Award.

## References

1. CCNx Synchronization Protocol http://www.ccnx.org/releases/latest/doc/technical/SynchronizationProtocol.html 2. Image source: World of Warcraft http://us.battle.net/wow/en 3. Image source: Quake http://www.quakeforge.net 4. Image source: Final Fantasy http://www.finalfantasyxiv.com 5. Unity Car Tutorial http://unity3d.com/support/resources/tutorials/car-tutorial