**Notes on the NDN-MOG**

jburke - January 19, 2013

Why are we doing an NDN-based game?

* Challenging performance requirements for networked multiplayer games.
* Widespread use (both in consumer environments and as simulations).
* Interest in 3D engine as a realtime compositing environment for a variety applications (REMAP past work).
* Would make for a good demo.

Requirements of the game decided *a priori*.

* Networked, multiplayer. To use NDN.
* Peer-to-peer. To challenge us to work without a centralized authority; because we think that NDN will be well-suited to making this structure work, which is otherwise very challenging in TCP/IP.
* Engine will be Unity3D. We have already invested in the software, have some initial development on NDN by Zening, and other projects that use it.
* Not doing a first person shooter. The world has enough of them.
* We use the story of Le Petit Prince to motivate the world. This was interesting to the artists and designers that we’re working with. But, we do not directly use characters and art for IP reasons.

Goals for NDN research

* Synchronize assets and state across multiple players with sufficient performance to meet typical player expectations and make game play.
* Make an “NDN-like” game that is influenced by the capabilities of the network and somewhat unique.

Why make a new game, instead of adapting an old one?

* More interesting for REMAP folks to get to do (at least a little) game research as well as networking research. Opportunity to create something novel on both ends.
* Don’t know of an existing game that would have the necessary set of qualities that are interesting.
* Don’t want IP constraints.

Key differences from Chen et al. 2012 (KK’s paper) –

* Actual implementation vs. simulation.
* Sync + interest/data over pub/sub library (at least so far)
* Approach driven by needs of an actual game, rather than a platform without a game (yet).

About *our game –*

* The work starts by focusing on player navigation of an interesting world first (ala TCG’s *Flow, Flower*) and interacting with it (or with non-player characters) then we will start to work on player-player interaction.
* We choose to “populate” the world based on real-world data, transformed into game objects or the state of game objects, to (1) have plenty of data to be synchronized; (2) provide a basis for interesting game play / navigation; (3) provide some experience in data visualization /simulation that might lead to future proposals and work; (4) be consistent with REMAP’s use of game engines.
* So for now, the game is just about moving from place to place and observing things, then we will build up interactions that change the world being observed.

Real-time data sources:

* FAA Airport Locations: These will be used to create “asteroids” and generate the universe of the world. From <http://flightwise.com/PlaneXML_API.html> To answer Lixia’s question – sync’ed between whom? – they will be sync’d between app box repos and then subsets sync’ed with game instances to generate “asteroid” locations.
* Real-time flight data (from airport to airport): These will create paths of travel and/or trace visualizations from airport-to-airport. (This borrow’s heavily from Aaron Koblin’s [*Flight Patterns*](http://www.aaronkoblin.com/work/flightpatterns/) to start, but only to begin.)
* Other local real-time data (perhaps from [Cosm](https://cosm.com/)) at each city, to create activity on the asteroids.
* Ilya has suggested stock market data, which is interesting, but it is not 24-hours and I’m having trouble translating it into the concept. Something to think about. <http://help.yahoo.com/kb/index?locale=en_US&y=PROD_FIN&page=content&id=SLN2310>

Three current pieces of the game architecture

* Data ingest and publishing. (From TCP/IP world to NDN world.) By Ilya.
* World visualization. (Reading data in NDN world and visualizing it in Unity). By Alex H, probably. This uses asset synchronization code by Zening.
* Player position synchronization across peers. (Show other players as avatars and ensure position in game is sync’d.) By Zening.

Data ingest and publishing

* Ilya will try to complete this quarter.
* Big picture idea is that there are *multiple, redundant samplers* that read data from the TCP/IP world and make it available in the NDN world. If any one fails, it should not affect availability of data in the system.
* Data is then synchronized (using a method tbd) across multiple repositories on the testbed.
* That is, multiple samplers feed multiple repositories.
* The samplers are implemented as PyCCN processes on app boxes throughout the testbed.
* So, there are a few parts to this:
  + Data ingest and publishing. (TCP/IP to repo storage, I guess?)
  + Repo sync and reconciliation. (Ilya new sync algo.)
  + Sync of game processes with this data. (Existing sync in Unity libraries for now?)
  + (+ Sync of graphical assets used to render the asteroids based on the data).

World visualization

* Alex H will likely work with Zening, Ren and Phoebe to create an initial visualization in Unity3D of the data, once given the namespace from Ilya.
* Uses asset sync code from Zening, hopefully, to pull sync game instances with data in the repository that is slowly changing. (from 11/25) <https://github.com/remap/ndn-lpp>
* Often-changing data will need other sync, right?

Player state sync

* Initially, there will no interaction between players, and the goal will be to provide perceptually synchronized player avatar position across peers.

Thoughts on scaling up P2P

* Games like this have a limited realm of player attention tied to their position in virtual geography.
* We can partition the game world into areas (I think the [KK paper](http://www.computer.org/csdl/proceedings/icdcs/2012/4685/00/4685a355-abs.html) does this) and synchronize only the relevant areas to a player.
* Going one step further, we should consider how to implement level-of-detail style functionality in synchronization of world data to each game instance, taking into account player *field-of-view* and *resolution*-*of-view***.** We do not need this for the first step (data ingest and sync) but it should be considered in the naming of that data, at least.

Responses to Zening email 1/17/13

I have been having these wonders for some time and I'd like to get some answers on:

(1) what exactly are our current interests in the NDN MOG project, SYNC algorithm or NDN MOG performance?

[jb] Hopefully the above notes answer this. At least initially.

(2) do we have an architecture preference, P2P v.s. C/S ?

[jb] P2P is an *a priori* requirement – we really want to see how well that would work.

**Our Interest**

I just realized that there is a big difference between (a) designing a game sync algorithm and (b) designing a game system that uses NDN, SYNC, some other sync algorithms and evaluate its performance.

The purpose of a game sync algorithm is to guarantee game state consistency. In its nature, a game sync algorithm sorts received packets according to their timestamp. If all packets are processed in the correct order, the game state would be consistent in all players' machines. If the algorithm is what we want to study, then we would be thinking about how to sort received packets, how to lock everybody's step to wait for late packets, or how to optimistically process received packets and rollback when the late packets arrived...

**I doubt this is what we want to do.** (Because this does not seem that closely related to NDN)

[jb] True. Insofar as we can apply an existing game synchronization algorithm (or set of algorithms) to our game, we do not need to re-implement. However, if our approach, being somewhat atypical in its nature, can’t use existing work, then it needs to be corrected. We want to make a somewhat unique game, so it may require a little bit of unique support.

On the other hand, if our primary interest is in the **performance of a synchronized NDN MOG**, then we would be considering the architecture of the game system, the scalability, network load and other performance-related issues. We can use those already proposed game sync algorithms. We can learn from game makers in the IP world, see what strategies they are using to scale their games up, and test if the performance is better in the NDN world.

**I think this is much more interesting and relavent. But it's also a much bigger topic**. The scope probably has to be narrowed down...

[jb] Yes, this sounds right. I would just point out that our interest is not soley in performance, but in the architectural approach and whether some hard features (e.g., P2P) are well supported by NDN.

**Architecture Preference**

This is based on the assumption that we are interested in the MOG system rather than the sync algorithm (see above) [er, I think you are talking about state reconciliation across instances, right, not exactly sync?]. Architecture choice matters for MOG performance, but not for sync algorithm -- as far as I understand it, game sync algorithms work for both P2P and C/S architectures, and there is no substantial difference in applying the algorithm to either architecture.

[jb] Well, I think we are interested in novel network protocols for sync, which may or may not require changes to the state reconciliation algorithms in the game. I suspect that as we try to scale the synchronization, peers may play a role in summarizing what’s going on with other peers, perhaps.

Game architectures come in P2P and C/S. Hybrids of the two exist, too. Nearly all MMOGs in the IP world and in the commercial world use C/S architecture. World of Warcraft (WoW) use cluster servers to host its millions of clients.

In academia, people believe that C/S architectures have some drawbacks, and many simulation work on P2P MMOGs have been done. In this paper ([Peer-to-peer support for massively multiplayer games](http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1354485&tag=1), Knutsson 2004), a P2P MOG of 4,000 concurrent players was simulated, and the performance looked fine.

However, there are skeptics. In ([The Near-Term Feasibility of P2P MMOG](http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5679578), Miller 2010), the simulation says that P2P MOGs are never going to scale given today's IP network and broadband bandwidth. (And I found it amusing :)

[jb] These are really interesting references. I’d like to suggest that we read them and discuss in a next meeting?

**MMOGs in the IP World**

I previously mentioned that there are many strategies that game designers in the IP world use to scale their games up and reduce the latency. The following paragraphs are about strategies that I have read about -- but it's in no way exhaustive.

*- P2P Overlay Networks and Application Level Multicast (ALM)*

In order to scale P2P MOGs up and limit latency, many P2P simulations use **overlay networks** and**ALM**. Overlay networks are often based on distributed hash tables (DHT). They limit the number of routing hops to *logN* (N is the number of players). So when the game scales to millions of plays, the latency is still bounded. ALM is used because of the lack of IP multicasting. ALM brings overhead and latency, and this is potentially a good thing for NDN! (The COPSS infrastructure is of course another alternative)

In our current game, players are interconnected by CCNx SYNC. I believe that a similar latency bound need to be established, once the game scales up.

[jb] Yes, keep in mind that our sync algorithm may evolve as Ilya’s research continues. Asset sync might continue to use existing algorithm, but I’m not sure. We should talk about it.

*- Distributed Computation of Game State*

This strategy works for both C/S and P2P. In C/S it may have names like "distributed scene graph", "sharding" or "replicated game state". Basically the global game state is duplicated or broken into smaller regional game states and hosted on server clusters. And the servers are synchronized with each other. In P2P, the global game state is broken into regional states or even entity states. Each peer hosts some of the states and all peers are synchronized with each other. Some state replication may also exist in P2P MOGS.

[jb] Yes, I’d like to design something that will work for the testbed app boxes and has an (unimplemented) idea for scaling.

*- Interest Management*

Both C/S and P2P MOGs need this. It is reported that in MMORPGs, players exhibit **Locality of Interest**. Players often only care about happenings that are within vicinity. Therefore, each player only needs to be informed updates that are within his/her **Area of Interest** (AoI). This is important for large scale games because it greatly reduces the number of players actually involved in synchronization. In a MMOG where there are 1 million players like the World of Warcraft, the largest interest group has about 40 players (when players gather to kill a "boss" monster). This greatly breaks down the hurdle of game making.

[jb] Yes, exactly. The interesting thing is that *what you are interested in the game is not related to your physical/topological location* (right?), which I think is intriguing. It means that there may be sync domains that are broken up by location, and perhaps have fuzzy boundaries (e.g., LOD implemented at the level of sync, perhaps). Something we should talk about.

Interest management seems intuitive in NDN, because we have names for data. So I guess it will be at least straightforward to implement :)

Response to Zening email 1/16/13

Recently I saw a lot of authors using **network traces** for simulation, e.g. using World of Warcraft traces as input to their simulator. I think this is a reasonable way to go :) The input is more realistic.

In this paper([MMORPG Player actions: Network performance, session patterns and latency requirements analysis](http://dl.acm.org/citation.cfm?id=1613039)), the authors invited six volunteers to play World of Warcraft and collected the network traces from the client's ends. These traces would be available upon request (for free, according to the paper). They may not be instantly useful, because there seems to be some encoding/decoding work to do, but they look interesting. **Shall I contact the authors?**

[jb] I am not sure yet about the relevance of that data to our problem, but if you want to take a look, I would by all means contact the author. I do think that we want to generate a legion of NPC that can help simulate use, when we get there, though. :)