Dennis Chan, dennischan@ucla.edu

Chih Chin Chang, changcc5@cs.ucla.edu

Jinda Zhuang, jindazhuang@gmail.com

**Proposal: Porting a turn-based Android game to NDN**

*I. Objective*

The objective of this project is to port an existing open-source, turn-based Android game - Domination (http://domination.sourceforge.net/) - to NDN in order to capitalize on NDN’s ad hoc capability. Domination is based on the popular board game Risk, in which two to six players occupy territories with their armies and attempt to win by capturing all the territories on the board. Capturing territories involves both the attacker and defender rolling dice to determine which side incurs casualties.

*II. Game Setup*

With NDN-based communication, there is no need for a centralized game server to handle the creation and joining of Domination games. Rather than the conventional lobby, each client attempts to maintain via synchronization a proper view of the list of games that are currently open. This synchronization may be implemented as a basic, simpler version of the process used in Chronosync (http://named-data.net/wp-content/uploads/2014/03/chronosync-icnp2013.pdf). When a game is created, it is named by a random string and appended to the list of open games along with the game settings, e.g. number of players. Each game also includes its creation time so that stale games can be pruned from the list. The game creator announces the prefixes for the open spots in his game. Clients can express interest in joining a particular game as a particular player, and the game creator is responsible for accepting players to the game by sending NDN data packets with the player name and position. Games are also removed from the list after all players have been accepted.

Without a centralized game server, there is a certain amount of additional game logic that needs to be moved to the clients. Tentatively, it is up to the game creator to initialize the board state. This involves using a random number generator to allocate the number of pieces for each player on the board. The sequence of play is also established randomly. There is an element of trust given to the game creator to avoid rigging the board at the outset, a problem which is not present in the traditional server model. Finally, once the board has been set up, the game creator announces the first turn of the game. The other players retrieve the data and set up their local boards accordingly.

*III. Taking Turns*

After a game is fully initialized, the players start the game by taking a move each turn based on the established sequence. In each term, the responsible player takes a move, a NDN data packet is then generated regarding the move for other players to sync the game board. For players waiting for their move would keep seeking the action data generated by the player taking a turn. Once a turn is completed, other players receive the actions taken by the responsible player and the game simulates the action and updates the actions in the game board for each player. The next player in sequence will then take over the responsibility of generating game update data for other players after taking a move.

In each turn, a sequence number is generated besides the game session ID created during initialization. Tentatively, the data packet name can then be generated in the form of applicationName/gameID/playerID/sequenceNumber/updates.

*IV. Ending the Game*

The end game introduces several interesting challenging to the NDN architecture. As we seek to explore solutions to these challenges, we first try to understand and define the procedures necessary to achieve an end game. From a single player’s perspective, the end of the game comes about when the player no longer holds any territory or holds every territory. So there are basically two challenges: how does a player recognize that the player has reached an end game and how can the player notify others that an end game has been reached.

Traditionally, when a game such as this is centrally hosted by one machine, that machine is responsible for updating and notifying the players, which makes these issues rather trivial. The NDN architecture can simulate such a host by having a single machine dedicated to these tasks. However, we believe that translating the traditional approach may not be the best solution. Therefore, we seek other designs; one where player communications can be sufficient to accomplish synchronization. One such design is to perform synchronization during the beginning of the turn phase, and players will poll for turns in regular time. Another idea is to reverse this and have the current player request interest to sync with others towards the end of the turn. Or to put it more simply, end games will simply be another action that will be handled by the turn phase. To implement the reverse interest, the current player would request an interest to end turn to each other player who will return an acknowledgement data and also send a request to update the board. When an interest to update the board is requested, the current player will provide the updated board.

*V. Further Considerations*

There are many potential issues that may arise in porting Domination to NDN. As mentioned earlier, it is possible for a client to purposely misrepresent the board state in its communications with other players. This is due to the lack of a centralized server that has no vested interest in the victor of any game and can be trusted to accurately relay game information. As such, it may be necessary at some point to enable each client to validate the board state in each turn. This does not fully resolve, however, the related issue of randomized dice rolls. In the same vein of trust, each client needs to maintain a certain level of security to prevent outside players from influencing the game. This can be achieved either by establishing some session password or presumably by utilizing NDN’s security features. Finally, not the last of these issues is the requirement that at least two players be playing at any time in order to properly synchronize the board state. In its existing form, Domination does not necessitate all players to have the application open at the same time because the game logic is stored server-side. Without such a server or permanent host, this drawback may be unavoidable.

Schedule of major steps:

Implement NFD to initialize and establish session between players: 4/24/2015

Create join game logic and initialize board state: 5/08/2015

Implement current turn phase game logic: 5/22/2015

Implement end of turn communication logic: 5/29/2015

Implement end of game cleanup: 6/05/2015

Catch errors and debug:

References:

ChronoSync: <http://www.named-data.net/techreport/TR008-chronos.pdf>

Named Data Networking Forwarding Daemon: <http://named-data.net/doc/NFD/current/overview.html>

Domination for Android: <http://domination.sourceforge.net>