NXNetwork: um *framework* para jogos em rede no iPhone

Eduardo Coelho, George Ruberti Piva,
Paulo César Rodacki Gomes, Dalton Solano dos Reis

[1]Departamento de Sistemas e Computação, FURB – Universidade Regional de Blumenau, Campus I, 89012-900, Blumenau, SC – Brasil

{eduardocoelho,piva,rodacki,dalton}@inf.furb.br

Abstract. *The recent release of Apple's iPhone SDK raised interest by the mobile game developers community, but this set of new technologies lacks further investigation. This paper aims to approach part of that technology by the proposal of NXNetwork, a client-server communication infrastructure for multiplayer games for iPhone using the Apple Bonjour framework.*

Resumo. *A recente publicação do iPhone SDK despertou o interesse da comunidade desenvolvedora de jogos para dispositivos móveis. Entretanto, trata-se de um conjunto de novas tecnologias que carecem de uma investigação mais aprofundada. O presente artigo procura abordar parte do problema propondo o NXNetwork, uma infraestrutura de comunicação cliente-servidor para jogos multiusuário na plataforma iPhone utilizando o framework Apple Bonjour.*

1. Introdução

Com crescente difusão do iPhone e a recente disponibilização do iPhone SDK pela Apple houve um aumento significativo no desenvolvimento de aplicações para esta plataforma. Através da App Store, tornou-se relativamente fácil a distribuição comercial de aplicações, o que acarretou no surgimento de um grande número de empresas especializadas na área bem como o interesse de universidades em estudar esta tecnologia. Recentemente, através do *iPhone Developer University Program*, tal fato tornou-se viável.

O iPhone possui características diferenciadas em relação as outros *smartphones* que são favoráveis ao desenvolvimento tanto de aplicações inovadoras quanto de games. Com base nisso, este artigo propõe uma infraestrutura para o desenvolvimento de jogos multiusuário em rede para esta plataforma utilizando o *framework* Bonjour. Para validar a proposta foi desenvolvido um jogo de AirHockey 2D. O jogo possui características que exploram o potencial do dispositivo bem como o iPhone SDK, uma vez que permite a jogabilidade entre dois jogadores simultaneamente, seja no mesmo aparelho (utilizando multitoque) ou em aparelhos diferentes (utilizando dos recursos de rede).

A próxima seção do artigo faz uma introdução ao iPhone SDK, a seguir são comentados o padrão Zero-configuration e o *framework* Bonjour. Na seção 5 é apresentado o jogo implementado utilizando o *framework* NXNetwork. A seção 6 descreve detalhadamente a estrutura deste *framework*. Na seção 7 são explicados os protocolos de conexão e comunicação implementados no *framework*. Por fim, a última seção apresenta os resultados obtidos.

2. O iPhone SDK

O iPhone SDK [Apple Computer 2008c] é uma plataforma que dá suporte ao desenvolvimento de jogos para as plataformas iPhone e iPod Touch. O iPhone OS é o sistema operacional desenvolvido pela Apple Inc. para estas plataformas e provê um grande número de funcionalidades úteis para o desenvolvimento de jogos, tais como: suporte a eventos multitoque, renderização 2D e 3D, acesso ao acelerômetro, serviços de rede, manipulação e persistência de dados, abstrações para a manipulação de janelas entre outras. O iPhone OS pode ser visto como um conjunto de camadas de abstração, conforme a figura 1.

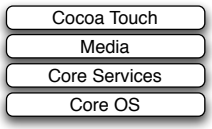

Figura 1. Camadas do iPhone OS

As camadas mais baixas do iPhone OS fornecem acesso aos serviços fundamentais do sistema, enquanto que camadas mais altas possuem serviços e tecnologias mais sofisticadas – são *frameworks* orientados a objeto que abstraem as camadas mais inferiores, mas que não necessariamente restringem o acesso a elas. O *Cocoa Touch* é uma das camadas mais importantes do iPhone OS, formada pelos *frameworks UIKit* e *Foundation* que provêm a infra-estrutura básica para desenvolvimento de aplicações gráficas e dirigidas a eventos. A camada *Media* provê os *frameworks* necessários para uso dos recursos de multimídia. Os *frameworks* de alto nível permitem a criação rápida de animações com recursos gráficos avançadoos, enquanto os *frameworks* de baixo nível provêm o acesso as ferramentas necessárias para desenvolver aplicativos customizáveis, a exemplo do *OpenGL ES*. A camada *Core Services* fornece os serviços fundamentais do sistema e, portanto, é utilizada por todas as aplicações direta ou indiretamente, esta camada contém os *frameworks Core Foundation* – responsável pelo gerenciamento básico dos dados e serviços do iPhone, *CFNetwork* – contém abstrações para protocolos de rede, entre outros. Por fim, a camada mais inferior do iPhone OS, a *Core OS* abrange o *kernel*, *drivers* e interfaces básicas do sistema operacional. É a camada responsável pelo gerenciamento de memória, *threads*, sistema de arquivos, rede e comunicação entre processos. Apenas um conjunto restrito de *frameworks* têm acesso ao *kernel* e *drivers* desta camada.

3. Zero-Configuration

O Bonjour é uma arquitetura de comunicação em rede baseada no padrão Zero-configuration, também conhecido como Zeroconf. Trata-se de um padrão da IETF para gerenciar uma rede TCP/IP sem a presença de um administrador ou qualquer configuração de rede. Tem como um de seus objetivos permitir que o usuário conecte seu computador/-dispositivo em uma rede local, e tenha acesso a todos os serviços nela disponíveis (tais como impressão, compartilhamento de dados e acesso a internet), não importando o tipo de conexão, Ethernet ou Wireless. Zeroconf é multiplataforma e é baseado somente nos protocolos TCP/IP como os de IP multicast e DNS [Voip-Info 2009]. Desde 2003 este padrão vem sendo suportado nos sistemas operacionais Mac OS, Microsoft Windows e Linux, e implementado em aplicativos e dispositivos [Cheshire 2009].

Para isso, no padrão atual do Zeroconf o sistema operacional ou dispositivo deve tratar três requisitos: (i) ser capaz de alocar para si, um endereço IP sem a presença de um servidor DHCP na rede (endereçamento); (ii) efetuar a tradução de nomes em endereços IP sem um servidor DNS (nomeação) e (iii) procurar por serviços na rede sem pedir a um servidor de serviços (descoberta de serviços). Seu desenvolvimento foi iniciado originalmente pela SUN Microsystems, porém seu uso tem sido promovido pioneiramente pela Apple Computer Inc. que o disponibiliza sob o nome Bonjour [Voip-Info 2009].

4. Bonjour

O Bonjour implementa as funcionalidades de endereçamento, nomeação e descoberta de serviços, definidos pelo padrão Zeroconf. Para o endereçamento, a solução proposta pelo Bonjour é a auto-atribuição de endereços IP em um intervalo reservado de endereços de uma rede local, tipicamente uma pequena LAN ou um segmento de LAN. Consiste em cada dispositivo escolher um endereço IP randômico e testá-lo. Se o endereço não estiver em uso, passa a ser seu endereço local, senão escolhe outro endereço randomicamente e tenta novamente. Com relação à nomeação, para a tradução nome-endereço em uma rede local, utiliza-se Multicast DNS (mDNS) onde requisições no formato DNS são enviadas para a rede local utilizando IP multicast. Desta forma não é necessário que se tenha um servidor DNS com conhecimento global para atender as requisições. Cada serviço ou dispositivo provê sua própria capacidade DNS - quando este recebe uma requisição com seu nome, responde-a com o seu endereço, uma vez que o *hostname* local tem significância apenas na rede local ou no segmento de LAN [Apple Computer 2008a].

O processo de nomeação é semelhante ao processo de endereçamento – cada serviço ou dispositivo atribui a si um nome e testa a sua disponibilidade. Se o nome do serviço a ser registrado pelo Bonjour não estiver disponível, ele renomeia o serviço por padrão. Um exemplo de um *hostname* local poderia ser: *Steve.local*. Por fim a descoberta de serviços permite que aplicações procurem instâncias de um tipo particular de serviço na rede e mantenham uma lista de nomes de serviços (a qual é persistente, ao contrário de endereços), podendo resolver um nome de serviço em um endereço e um número de porta quando necessário. A descoberta de serviços é feita através do envio de uma requisição mDNS para um tipo de serviço e domínio, onde cada serviço compatível responderá com seu nome, resultando em uma lista serviços disponíveis.

Diferentemente da abordagem tradicional orientada a dispositivos, o Bonjour é orientado a serviços. Isto faz com que não seja necessário que se requisite a todos os dispositivos "que serviços estes fornecem" e sim, "quais fornecem determinado serviço", evitando tráfego desnecessário na rede. Para nomeação de instâncias de serviços Bonjour, utiliza-se a convenção: *"_nomeServiço._tipoServiço._nomeProtocoloDeTransp.domínio"*. Um exemplo de nome válido para uma instância de um serviço Bonjour é *"I-601's iPhone._airhockeygame._tcp.local."*, o qual representa um serviço do tipo *airhockeygame* que utiliza o protocolo de transporte TCP publicado no dominio *local.*, o primeiro nome é uma string visível ao usuário que indica o nome do serviço. Cada elemento é separado pelo caracter '_'. A arquitetura de serviços do Bonjour suporta três operações básicas: publicação e descoberta de serviços e resolução de endereços. A figura 2 exemplifica a publicação de um serviço de um servidor de jogo de AirHockey oferecido por um iPhone na rede local, onde o protocolo de transporte utilizado é o TCP e o protocolo de aplicação é denominado *airhockeygame*.

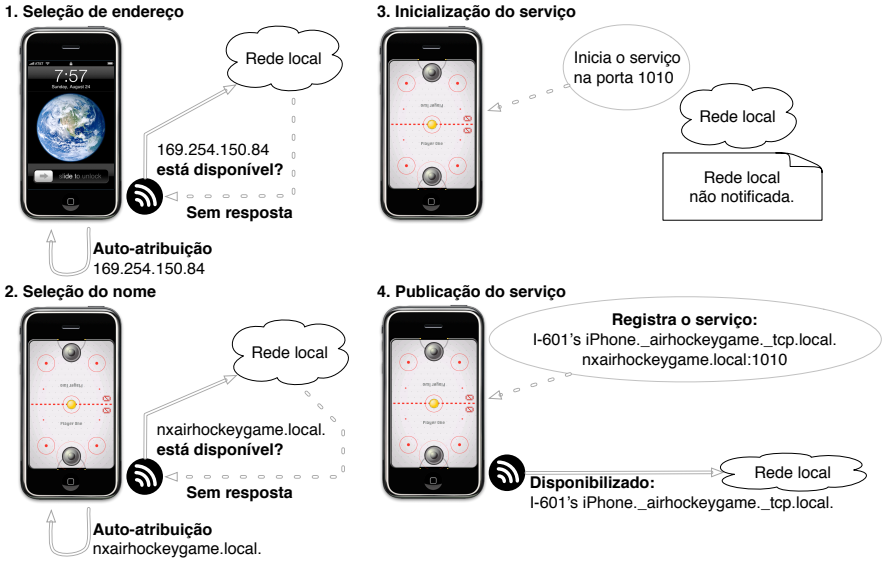

Figura 2. Publicação de um serviço de um servidor de jogo de AirHockey

A figura 3 ilustra uma aplicação cliente fazendo uma busca por todos os serviços de servidores de jogo de AirHockey disponíveis na rede local. A figura 4 demonstra a resolução de uma instância de um serviço de um servidor de jogo de AirHockey.

Figura 3. Buscando serviços de um servidor de jogo de AirHockey

O SDK da Apple disponibiliza uma série de APIs para aplicações de serviços Bonjour, dentre as quais se destaca as classes *NSNetService e NSNetServiceBrowser*, presentes no *Foundation framework*. Estas classes provêm abstrações orientadas a objetos para a publicação e a descoberta de serviços. As tarefas realizadas por objetos destas classes ocorrem assíncronamente, comumente em uma *thread* diferente da *thread* principal da aplicação, reportando os resultados das operações aos seus objetos delegados[1].

5. O jogo NXAirHockey

Com o intuito de testar o *framework* em questão, foi desenvolvido um jogo de AirHockey 2D multiusuário utilizando o motor de jogos NXEngine. Este motor fornece subsídios para desenvolvimento de jogos 2D multiusuário com suporte a interface multitoque. Os recursos para jogo multiusuário são implementados pelo *framework* NXNetwork.

[1]Um objeto delegado é aquele que atua em favor de (ou em coordenação com) outro objeto quando este recebe um evento no programa[Apple Computer 2008b].

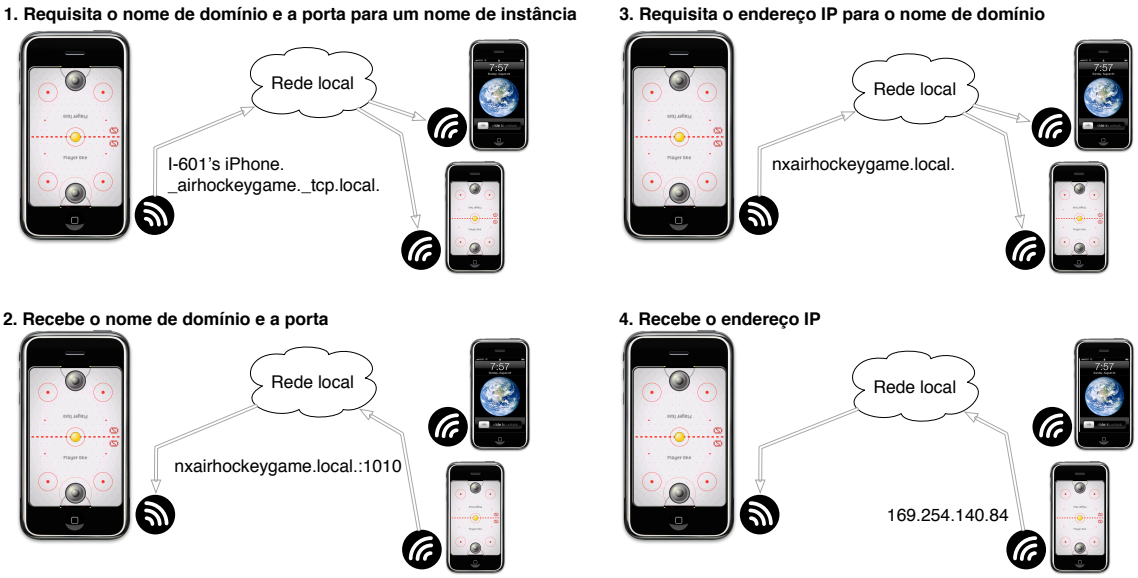

Figura 4. Resolução de instância de serviço

O AirHockey é um jogo para dois jogadores os quais têm como objetivo marcarem pontos atingindo o gol adversário [Wikipedia 2009]. Na versão implementada para iPhone/iPod Touch, denominado NXAirHockey, é possivel jogar no modo monousuário (contra o computador) ou no modo multiusuário. No modo multiusuário pode-se jogar em um mesmo dispositivo, utilizando os recursos de multitoque, ou então, em diferentes dispositivos através da comunicação *wireless*, que é o foco deste artigo. A figura 5 apresenta o jogo no modo multiusuário sendo executado em diferentes dispositivos juntamente com os principais componentes do NXAirHockey.

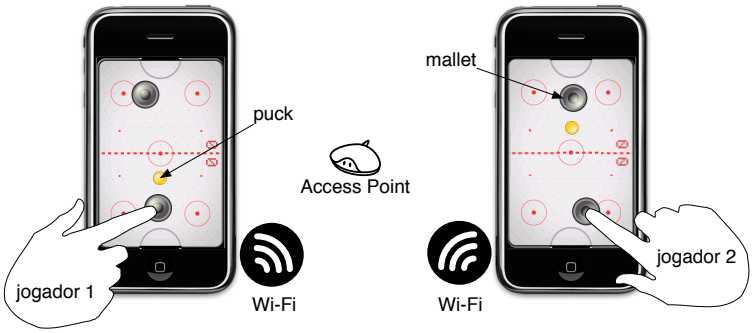

Figura 5. Jogo NXAirHockey em modo multiusuário.

6. O *framework* NXNetwork

Este *framework* provê uma estrutura básica para criação de jogos multiusuário utilizando comunicação cliente-servidor juntamente com as classes NSNetService e NSNetServiceBrowser. Os protocolos que compõem o *framework* são apresentados na figura 6, juntamente com classes que abrangem o motor de jogos e o jogo propriamente dito. Os protocolos do *framework* são compostos da seguinte forma:

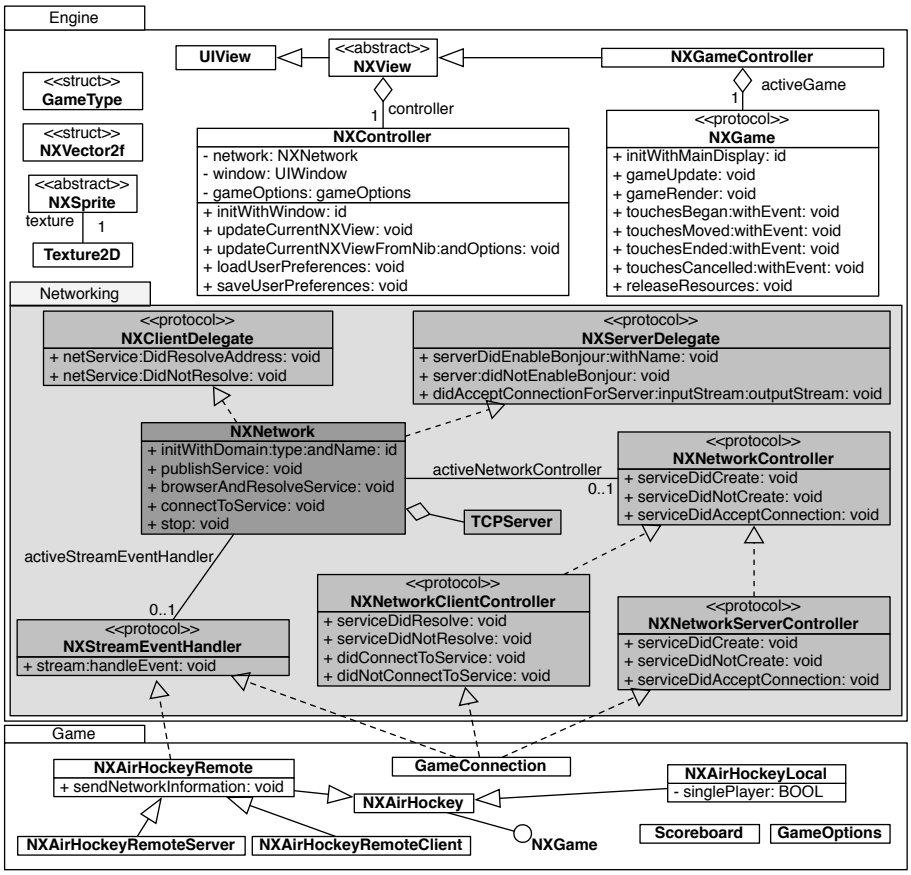

Figura 6. Diagrama de classes do jogo NXAirHockey e do motor NXEngine

- **NXServerDelegate:** este protocolo define as operações *serverDidEnableBonjour:withName:*, *server:didNotEnableBonjour:* e *didAcceptConnectionForServer:inputStream:outputStream:* que devem ser implementadas por uma classe delegada para gerenciar a criação de um servidor e a publicação de um serviço Bonjour. Estas operações são invocadas quando um serviço Bonjour foi publicado com sucesso, um serviço não foi publicado ou quando um serviço recebeu uma conexão, respectivamente;
- **NXClientDelegate:** protocolo que define as operações *netService:DidResolveAddress:* e *netService:DidNotResolve:* que devem ser implementadas por uma classe delegada para gerenciar a resolução de um serviço Bonjour. Tais operações são invocadas quando um serviço Bonjour foi ou não resolvido, respectivamente;
- **NXNetworkController:** define a operação *setupNXNetwork:* que deve ser implementada por uma classe que deseja controlar um objeto NXNetwork. Através da operação descrita, deve-se inicializar o objeto controlado e informar ao mesmo que a classe que adota este procolo é sua controladora;
- **NXNetworkServerController:** especialização do protocolo NXNetworkController que define as operações *serviceDidCreate:*, *serviceDidNotCreate:* e *serviceDidAcceptConnection:*, as quais devem ser implementadas para controlar o processo de publicação de um serviço Bonjour. As operações descritas são invocadas pelo objeto controlado, respectivamente,

quando um serviço foi ou não publicado ou quando foi estabelecida uma conexão para o mesmo, permitindo a tomada de decisões pela classe que adota este protocolo;

- **NXNetworkClientController:** especialização do protocolo NXNetworkController que define as operações *serviceDidResolve:*, *serviceDidNotResolve:*, *didConnectToService:* e *didNotConnectToService:*, as quais devem ser implementadas para controlar o processo de resolução e conexão a um serviço Bonjour. São invocadas pelo objeto controlado, respectivamente, quando um serviço Bonjour foi ou não resolvido bem como quando uma conexão a um serviço previamente resolvido foi ou não estabelecida;
- **NXStreamEventHandler:** protocolo responsável por definir a operação *stream:handleEvent:* que controla o fluxo de *bytes* através dos *streams* de entrada e saída presentes em uma instância de NXNetwork. Os *streams* são criados após ter sido estabelecida uma conexão a um serviço Bonjour.

6.1. A classe NXNetwork

Havendo comumente uma única instância na aplicação, esta classe adota os protocolos NXServerDelegate e NXClientDelegate podendo portanto publicar, descobrir ou resolver um serviço Bonjour, representado pelo atributo *netService* que é um objeto da classe NSNetService. Uma vez definidos o domínio, o nome e o tipo de serviço e o protocolo de transporte, é possível, através das operações *publishService:* e *browseAndResolveService:* realizar as três operações básicas do Bonjour:

- publicação: instancia-se um objeto da classe TCPServer – responsável pela criação do servidor TCP – e publica-se o serviço através da operação *publish:* do atributo *netService*;
- descoberta de serviços: utiliza-se um objeto da classe NSNetServiceBrowser – responsável por retornar todos os serviços do tipo especificado – e atualiza-se o atributo *netService* com o serviço desejado;
- resolução: chama-se a operação *resolveWithTimeout:* do atributo *netService*.

Durante a execução das operações acima, os eventos reportados são notificados ao atributo *activeNetworkController* – uma referência a um objeto que adota o protocolo NXNetworkController – o qual deve ser informado através da operação *setActiveNetworkController:*. Após ser estabelecida uma conexão através da operação *connectToService:*, esta classe é responsável por abrir os *streams* de entrada e saída, que são instâncias de NSInputStream e NSOutputStream, além de repassar os eventos gerados por esses objetos para o atributo *activeStreamEventHandler*. Este atributo é uma referência a um objeto que adota o protocolo NXStreamEventHandler, o qual deve ser informado através da operação *setActiveStreamEventHandler:* para a transmissão de *bytes* na rede. Por fim, a operação *stop:* encerra os serviços ativos no momento.

7. Protocolo de Comunicação

Os protocolos de conexão do jogo NXAirHockey nos modos cliente e servidor são apresentados nos diagramas de sequência da figura 7. Seguindo a arquitetura do motor de jogos NXEngine, para que uma aplicação cliente estabeleça uma conexão a um serviço Bonjour referente a um servidor do jogo NXAirHockey, ela deve requisitar ao objeto NXController a instância única de NXNetwork na aplicação e em seguida inicializá-la através da

operação *setupNXNetwork:*. Na seqüencia é realizado o processo de descoberta, resolução e conexão com o serviço Bonjour para que sejam obtidos os *streams* de entrada e de saída. Em seguida, a aplicação cliente requisita o início do jogo através da mensagem *"get srv"*, obtendo como resposta a mensagem *"srv ok"* e então inicia-se o jogo. O processo é semelhante com relação a uma aplicação servidora do jogo NXAirHockey, porém, caso não seja encontrado nenhum serviço do tipo *_airchokeygame._tcp.local.*, a aplicação cria uma instância de TCPServer e publica o serviço Bonjour no domínio local. Na seqüencia, o servidor aguarda uma requisição até que seja feita a troca de mensagens e então iniciado o jogo.

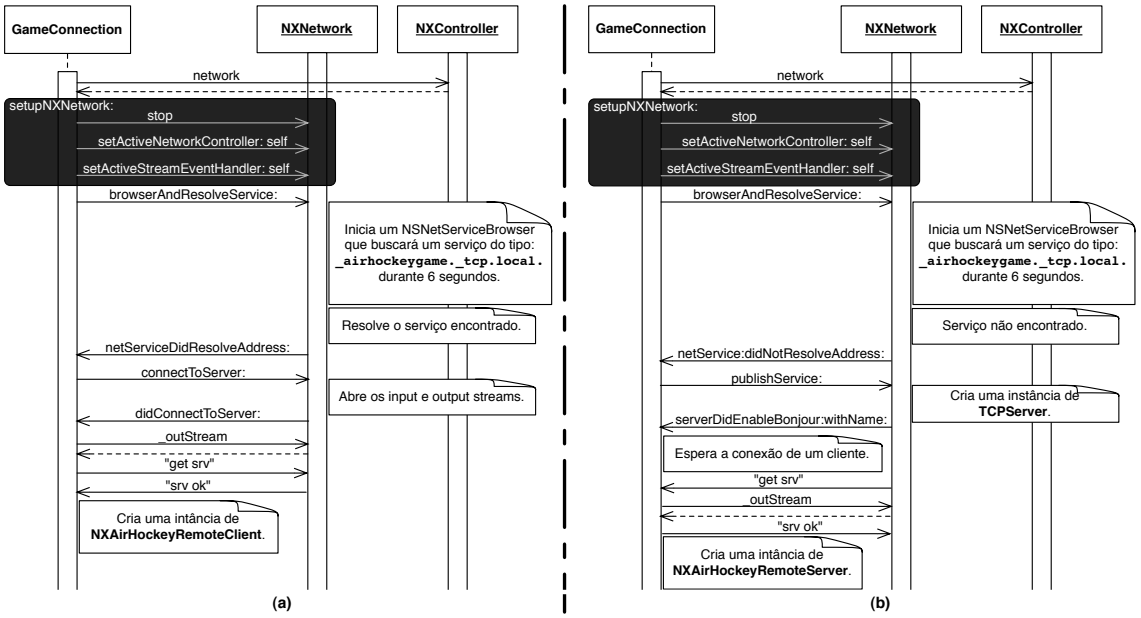

Figura 7. Protocolo de conexão nos modos (a) cliente e (b) servidor

A implementação da operação *stream:handleEvent:* na classe GameConnection é apresentada no algoritmo 1. Após estabelecida a conexão, a comunicação entre as instâncias de NXAirHockeyRemoteServer e NXAirHockeyRemoteClient é feita conforme o diagrama de sequência da figura 8.

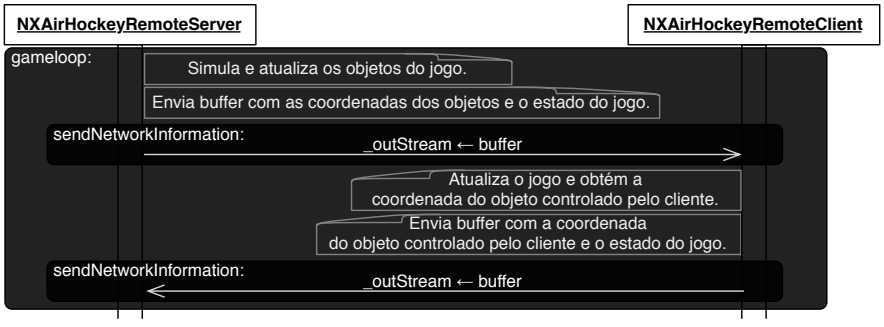

Figura 8. Protocolo de comunicação entre cliente e servidor do NXAirHockey

O servidor é responsável por simular todos objetos do jogo e, através da operação *sendNetworkInformation:* enviar as coordenadas dos mesmos - exceto as coordena-

```
Entrada: Um stream (da classe NSStream) e streamEvent (da classe NSStreamEvent)
1  NSInputStream inStream ← controller.network.inStream;
2  selecione streamEvent faça
3      caso NSStreamEventEndEncountered
4          Encerra jogo;
5      caso NSStreamEventHasBytesAvailable
6          se (stream == inStream) e (inStream possuir bytes disponíveis) então
7              int tamanho ← tamanho do buffer de entrada;
8              inBuffer ← leitura do buffer de entrada;
9              se tamanho > 0 então
10                 se inBuffer == "get srv" então
11                     outBuffer ← "srv ok";
12                     Envia outBuffer para o stream de saída;
13                     Inicia jogo em modo sevidor;
14             senão
15                 se inBuffer == "srv ok" então
16                     Inicia jogo em modo cliente;
17     caso OutroCasoATratar
18         Trate o outro caso...;
```

Algoritmo 1: Tratmento de evento na classe de conexão

das do *mallet* do cliente - para a aplicação cliente. O cliente recebe um evento e através da operação *stream:handleEvent:* obtém o *buffer*, atualiza as coordenadas dos objetos do jogo e envia as coordenadas de seu *mallet* para o servidor para que seja feita novamente a simulação, formando assim o laço principal do jogo. Um parâmetro adicional que representa o estado do jogo (pausado, etc) é concatenado aos *buffers* enviados. A implementação da operação *sendNetworkInformation:* na classe NXAirHockeyRemoteServer é apresentada no algoritmo 2.

```
1  Inicializa outBuffer;
2  outBuffer ← posição do puck no jogo servidor;
3  outBuffer ← posição do mallet no jogo servidor;
4  outBuffer ← estado atual do jogo;
5  Escreve outBuffer no stream de saída;
```

Algoritmo 2: Tratmento de evento na classe de conexão

Um exemplo de código para realizar esta operação pode ser visto abaixo. As operações *stream:handleEvent:* nas classes NXAirHockeyRemoteServer e NXAirHockeyRemoteClient, e a operação *sendNetworkInformation:* na classe NXAirHockeyRemoteClient são implementadas de forma semelhante.

```
- (void) sendNetworkInformation {
    // prepara o buffer com a do mallet do cliente (playerTwo) e o estado do jogo
    uint8_t _outBuffer[16];
    sprintf(_outBuffer, ``%3.0f %3.0f %c'',
            [playerTwo position].x, kFieldHeight-[playerTwo position].y, nxbitmask);

    // escreve o buffer no stream de saída
    [[[[gameController controller] network] _outStream] write:_outBuffer
                                                    maxLength:sizeof(_outBuffer)];
}
```

8. Considerações finais

Este artigo apresentou o desenvolvimento do NXNetwork, um *framework* para jogos em rede multiusuários para iPhone. O NXNetwork é fortemente dependente das tecnologias fornecidas pelo Apple iPhone SDK, notadamente o Bonjour, que facilita a comunicação entre os dispositivos através de Wi-fi.

O Bonjour dá suporte aos protocolos TCP/IP e UDP/IP. Normalmente a comunicação em rede para jogos no estilo do NXAirHockey implementado seria mais adequada com o uso do UDP/IP [Chen et al. 2006] porque este protocolo transmite pacotes menores. Ao contrário, o TCP/IP já apresenta um cabeçalho maior que o própria conteúdo típico de uma mensagem transmitida no NXNetwork. Além disso o TCP/IP garante a entrega de pacotes, o que para a transmição contínua de coordenadas não seria necessário. Entretanto, para o caso de eventos essenciais tais como marcação de gol ou solicitação de pausa na aplicação, a garantia de entrega de pacotes TCP/IP é útil. Além disso, o uso de UDP/IP com o Bonjour está disponível apenas em camadas de baixo nível no iPhone SDK. Na implementação atual do NXNetwork foram utilizadas as classes descendentes de NSStream presentes na camada Foundation.framework do iPhone SDK para a transmissão de *streams* de *bytes*. Estas classes possuem boa documentação e exemplos disponíveis no site da Apple.

Os resultados foram satisfatórios onde, em condições normais de jogo, foi possível jogar no modo multiusuário em diferentes dispositivos sem diferenças significativas com relação ao modo monousuário e com taxas na média de 60 *frames* for segundo, tanto no simulador quanto no dispositivo.

9. Agradecimentos

Os autores agradecem à Universidade Regional de Blumenau e ao Projeto ACREDITO do curso de Bacharelado em Ciências da Computação da FURB pelo apoio financeiro para aquisição de equipamentos para realização do trabalho.

Referências

Apple Computer (2008a). *Bonjour Overview*. Apple Computer Inc, Disponivel em: http://developer.apple.com.

Apple Computer (2008b). *Cocoa Fundamentals Guide*. Apple Computer Inc, Disponivel em: http://developer.apple.com.

Apple Computer (2008c). *iPhone OS Programming Guide*. Apple Computer Inc, Disponível em: http://developer.apple.com.

Chen, K.-T., Huang, C.-Y., Huang, P., and Lei, C.-L. (2006). An empirical evaluation of tcp performance in online games. In *Proceedings of ACM SIGCHI ACE 06*, Los Angeles USA.

Cheshire, S. (2009). *Zero Configuration Networking (Zeroconf)*. Disponivel em: http://www.zeroconf.org/.

Voip-Info (2009). *Asterisk Zeroconf Support - voip-info.org*. Disponivel em: http://www.voip-info.org/wiki/view/Asterisk+Zeroconf+Support.

Wikipedia (2009). *Air Hockey - Wikipedia*. Disponivel em: http://en.wikipedia.org/wiki/Air_hockey.