VFC Android : Consistência em redes móveis

Mário Santos

INESC-ID/IST
mario.santos@ist.utl.pt

Abstract. Na sociedade de hoje, os telemóveis fazem parte da vida quotidiana de qualquer pessoa e é difícil encontrar alguém que não tenha um dispositivo deste género. Estes dispositivos são usados para efeitos pessoais como para efeitos de lazer como jogos, músicas, fotografias, etc.... Os jogos multi-jogador para estes dispositivos exigem uma grande comunicação entre eles, necessária para manter o estado do jogo consistente entre os jogadores. O objectivo deste trabalho é implementar um modelo de consistência Vector Field Consistency (VFC) baseado em técnicas de Gestão de Interesse, na plataforma Android, e desenvolver um jogo que demonstre as suas vantagens para ambientes móveis em redes ad-hoc. A redução do número de mensagens necessárias para alcançar um grau aceitável de consistência vai proporcionar um aumento na autonomia do dispositivo, sem afectar a jogabilidade do jogo.

Keywords: Dispositivos Móveis, Ad-Hoc, Consistência, Gestão de Interesse

1 Introdução

Os telemóveis são dispositivos que têm vindo a ganhar uma grande importância na sociedade. Nos dias de hoje qualquer pessoa possui um ou mais dispositivos deste género usando-os para trabalho ou para lazer. Nos últimos anos a performance do hardware destes dispositivos tem vindo a crescer quase exponencialmente. Se antes um telemóvel era um dispositivo unicamente destinado para trocar chamadas ou sms, hoje é um dispositivo altamente avançado possibilitando aos utilizadores o uso de aplicações avançadas, como a navegação na internet, integração de um GPS ou mesmo simples jogos.

Os jogos multi-jogador para estes dispositivos começaram a aparecer nos últimos anos com a introdução de novas interfaces de rede como o Bluetooth[10] ou o WiFi[1]. Este trabalho foca-se nestes jogos tendo como suporte uma rede ad-hoc. Supondo um cenário em que existem duas pessoas com dispositivos deste género, elas podem competir entre si num jogo multi-jogador em qualquer lugar, seja num autocarro, num centro comercial ou em casa desde que estejam relativamente próximos um do outro. O ambiente ad-hoc torna possível a execução destes jogos sem recorrer a quaisquer infra-estruturas auxiliares; apenas são necessários os telemóveis dos jogadores.

Para qualquer tipo de jogos que impliquem vários jogadores, tendo cada um o seu dispositivo, é necessário manter a consistência entre os dois dispositivos. Esta consistência é necessária para que os vários jogadores tenham a mesma percepção do estado do jogo, mantendo uma jogabilidade agradável. A consistência entre os dois dispositivos é mantida através da troca de mensagens entre eles. Estas mensagens são eventos do jogo que acontecem nos dispositivos e que necessitam de ser difundidas para conseguir manter a lógica do jogo o mais actualizada possível. A consistência acaba por estar directamente relacionada com o número de mensagens, pois se queremos ter os dois dispositivos o mais sincronizado possível, tem que existir um grande número de mensagens a serem transmitidas.

Uma excessiva troca de mensagens entre os dois dispositivos pode levantar problemas. As interfaces de rede existentes nestes dispositivos representam um grande consumo de bateria neste tipo de aplicações. A recepção de mensagens implica o processamento das mesmas o que corresponde a um maior consumo da bateria.

A latência é um problema associado a qualquer aplicação distribuída seja ela um jogo ou não. As técnicas existentes conseguem mascarar a latência existente nas redes, mas não a conseguem eliminar. Actualmente são usadas técnicas de atribuição de prioridades aos pacotes em redes congestionadas, atrasos aplicados no lado do cliente e técnicas de Dead Reckoning. O Dead Reckoning é uma técnica que mascara a latência do lado do jogador, tentando prever qual a próxima localização dos objectos no mapa.

A Replicação Optimista é um modelo de consistência muito usado em jogos multi-jogador. Através da Replicação Optimista conseguimos reduzir o número de mensagens necessárias para obter consistência em dados replicados entre os jogadores. Ao contrário de modelos Pessimistas, a Replicação Optimista permite que os jogadores possam ler dados inconsistentes.

Os mecanismos de Gestão de Interesse são fundamentais em jogos multi-jogador. A Gestão de Interesse funciona como um filtro do lado do Jogador. Um jogador apenas vai receber eventos que realmente lhe interessem, deixando de parte eventos irrelevantes. Este filtro de eventos só é aceite pelo jogador se isso não causar um grande impacto na sua jogabilidade. A proximidade entre objectos e o jogador é uma maneira de especificar o nível de interesse do jogador.

Os sistemas existentes que aplicam técnicas Gestão de Interesse em jogos multi-jogador, não estão focados para as limitações que os telemóveis apresentam, a autonomia limitada da bateria e a fraca capacidade de processamento. Os sistemas mostram ser pouco flexíveis e alguns estão orientados para um tipo de jogo específico, não permitindo a aplicação do sistema em jogos de outro tipo.

O objectivo deste trabalho é implementar um modelo de consistência VFC baseado em técnicas de Gestão de Interesse, na plataforma Android, e desenvolver um jogo que demonstre as suas vantagens para ambientes móveis em redes ad-hoc. Este sistema vai permitir reduzir o tráfego gerado entre os vários jogadores, o que corresponde num menor consumo de energia dos dispositivos móveis. Este sistema vai servir como uma ferramenta simples para os programadores de jogos multi-jogador usarem e que os vai abstrair de todo o processo

de manutenção de consistência, entre os estados dos vários jogadores existentes na rede.

Este relatório tem a seguinte organização: na secção 2 iremos abordar o trabalho relacionado nesta área, focando-nos nas técnicas e sistemas mais usados; na secção 3 pode ser encontrada a descrição do Algoritmo e Arquitectura da solução; a secção 4 contem a metodologia de avaliação do trabalho e por último na secção 5 encontra-se a conclusão.

2 Trabalho Relacionado

Nesta secção vamos abordar várias técnicas usadas para redução do impacto de latência e redução do número de mensagens necessárias para a manutenção da consistência em jogos multi-jogador. Serão também apresentados vários sistemas existentes que aplicam estas técnicas. Por último, será apresentado o trabalho relacionado sobre dispositivos móveis e redes ad-hoc.

2.1 Redução do impacto da Latência

A latência entre dois nós de uma rede é um dos maiores problemas existentes nos jogos multi-jogador. A existência de latência significa que as mensagens levam um certo tempo até chegarem aos seus receptores. A latência pode originar situações paradoxais como, por exemplo, num jogo de corridas haver 2 jogadores em primeiro lugar ao mesmo tempo. As alternativas existentes[17, 18] não conseguem eliminar o problema da latência. Actualmente, o que se faz é aplicar técnicas que reduzem o impacto causado pela latência.

Uma das técnicas[17] é o atraso na apresentação local ao jogador, o que implica que os eventos sejam atrasados numa fila de eventos. Outra das técnicas existentes[18] é o uso de mecanismos para antecipação de eventos. A posição do jogador altera-se no decorrer de um jogo. É possível prever a posição futura do jogador, com algum intervalo de confiança, tendo como base a sua posição actual e as posições anteriores.

Atraso da apresentação local O atraso na apresentação local do estado do jogo[17] ao utilizador consegue reduzir o impacto da latência num jogo multi-jogador. Nesta técnica, os eventos produzidos pelos jogadores não são aplicados de imediato no estado do jogo. Os eventos são atrasados numa fila de eventos que são aplicados ao estado do jogo ao fim de um pequeno intervalo de tempo. Este intervalo de tempo tem de ser o mínimo possível, pois um grande atraso na aplicação dos eventos pode notar-se na jogabilidade deixando o jogador insatisfeito. Durante este intervalo de tempo, são recebidos todos os eventos de todos os jogadores de maneira a garantir que todos vão aplicar os mesmos eventos, pela ordem correcta, e vão ter um estado do jogo consistente entre eles. Para garantir a recepção de todos os eventos o intervalo de tempo tem de ser igual à latência máxima da rede, caso contrário, podem faltar eventos na fila de eventos.

A eficácia desta solução depende muito do tipo de jogo a que é aplicada. No caso de jogos que precisem de uma jogabilidade de resposta rápida, como é o caso dos jogos de corridas, ou jogos de tiros, esta solução tem menos sucesso pois o intervalo de tempo tinha que ser muito pequeno. Outra desvantagem desta solução é a possibilidade de ocorrer variância nas latências. No caso das redes sem fios, as latências podem variar bastante, como por exemplo quando aparece um obstáculo entre os dispositivos, sendo difícil definir um valor para o intervalo de tempo.

Dead Reckoning O Dead Reckoning[18] é uma técnica usada para reduzir o impacto da latência da rede, através da antecipação de futuros pacotes. Com isto conseguimos reduzir o impacto do atraso dos pacotes, bem como a perda dos mesmos. Contudo, esta antecipação necessita de ser feita com um grande rigor, para não originar grandes discrepâncias entre o evento estimado e o evento real.

Os métodos de previsão de eventos de localização tentam antecipar a posição futura de um objecto, baseando-se na sua posição actual e nas posições anteriores. Para este cálculo usam-se dados como direcção, velocidade, aceleração e polinómios. O uso de polinómios permite uma melhor modelação caso a trajectória descreva uma curva, tornando a previsão mais correcta. Após a chegada dos pacotes esperados, é feita uma comparação entre a posição estimada e a posição correcta. Havendo diferenças é necessário corrigir a posição do jogador. Uma solução básica para este problema é colocar de imediato o objecto na posição correcta o que pode originar saltos repentinos, levando a uma diminuição da jogabilidade. Existem soluções mais avançadas que tentam convergir a posição do jogador usando uma convergência linear. Uma convergência linear escolhe vários pontos entre as duas posições. Com este método, a convergência entre os dois pontos é feita de uma maneira mais suave, reduzindo assim o impacto negativo na jogabilidade.

Uma das vantagens desta técnica é a redução na frequência de mensagens enviadas. Durante a ausência das mensagens são aplicas as técnicas de antecipação tentando assim compensar a falta das mesmas. A desvantagem desta solução é que depende muito da qualidade da previsão que é feita. Os sucessos das previsões são muito dependentes do tipo de jogo e do modo como o jogador joga.

Em jogos de corridas é mais fácil prever qual vai ser a próxima posição do jogador. Isto acontece porque um jogador normalmente segue a trajectória da pista e as posições tendem a repetir-se em todas as voltas à pista. Em jogos como First Person Shooters torna-se mais difícil antecipar a próxima acção do jogador devido à quantidade de acções que um jogador pode efectuar em qualquer momento.

O modo como o jogador joga também influencia as previsões. As acções dum jogador que jogue mais bruscamente e que faça movimentos rápidos, são mais difíceis de prever, que as acções de um jogador que jogue de uma maneira mais suave e lenta.

2.2 Mecanismos ao nível da rede

Existem várias soluções aplicadas ao nível da rede que reduzem a latência e a largura de banda necessária para o fluxo de pacotes pelos nós. Estas soluções são aplicáveis a qualquer tipo de aplicações distribuídas e, em particular, a jogos multi-jogador. Duas soluções interessantes nesta área são: a 1) atribuições de um nível de urgência e relevância aos pacotes e 2) agregação e compressão dos pacotes.

Atribuição de Prioridades aos pacotes A atribuição de uma urgência e relevância dos pacotes [9] permite mascarar a latência introduzida devido ao congestionamento dos pacotes na rede. Pacotes com uma urgência mais elevada são enviados primeiro que pacotes menos urgentes, garantindo que chegam ao seu destino no menor tempo possível, reduzindo assim a latência introduzida por congestionamentos na rede. Os pacotes menos urgentes vão sendo atrasados até já não haver pacotes urgentes para enviar. A relevância atribuída aos pacotes é uma medida de fiabilidade muito útil quando aplicada a protocolos que não garantam qualquer tipo de fiabilidade como, por exemplo, o protocolo UDP. Um pacote com uma relevância alta significa que será retransmitido várias vezes até se confirmar a recepção no seu destino. Isto significa que existe um contador de reenvio de mensagens que se vai decrementando sempre que a mensagem não chega ao seu destino. Quando um pacote tem uma relevância baixa, significa que a sua perca não vai ter um impacto grande no jogo.

Esta solução tem duas grandes desvantagens quando aplicada a jogos multi-jogador. Em primeiro lugar não tem em conta qualquer tipo de localidade das entidades do jogo, ou seja, não são considerados factores como, por exemplo, a distância entre as duas entidades. Outra desvantagem é o facto de a atribuição ter que ser feita na fase de desenho do jogo. A urgência e relevância dos eventos estão incorporadas nas classes dos eventos. Desta maneira não é possível alterar a urgência e relevância de um evento durante a execução do jogo. Esta desvantagem tem um grande impacto num jogo multi-jogador, em que um evento pode ter urgências diferentes consoante o estado do jogo.

Compressão e Agregação de pacotes Actualmente existem técnicas de compressão e agregação de pacotes [24] que permitem reduzir a largura de banda necessária para a transferência de pacotes.

A ideia base da compressão é conseguir transmitir a mesma mensagem usando o menor número de bytes possível. Nas técnicas de compressão a mensagem inicial é comprimida no emissor, enviada ao receptor e por fim descomprimida, obtendo assim a mensagem inicial enviada pelo emissor. As mensagens, como vão comprimidas, ocupam menos bytes e são transmitidas mais rapidamente. A desvantagem desta solução é custo associado aos passos de compressão e descompressão adicionados no emissor e receptor, introduzindo mais computação e tempo.

A agregação de pacotes consiste no agrupamento de pacotes num só pacote. Desta maneira em vez de enviar vários pacotes pequenos, os pacotes são agregados num só. Esta solução reduz a largura de banda necessária reduzindo os cabeçalhos dos pacotes IP. Em vez de ter n cabeçalhos de n mensagens, como os pacotes são agregados num só, é utilizado apenas um cabeçalho. A agregação pode ser feita com base num temporizador ou num contador. Em agregações com um temporizador os pacotes são agregados até expirar um limite temporal. Em agregações com contador, os pacotes são agregadas até acumular n pacotes.

A agregação com base num temporizador tem como maior desvantagem a introdução de mais latência na rede, pois os pacotes agora só são enviados após expirar um limite temporal. Agregação com base num contador tem como desvantagem o intervalo de tempo até atingir n mensagens. Como só são enviados pacotes até acumular n pacotes corremos o risco de esperar muito tempo até acumular os n pacotes.

2.3 Modelos de Consistência

A replicação de dados consiste em manter múltiplas cópias de dados, chamados réplicas, em computadores distintos. Com a replicação conseguimos ter maior disponibilidade no acesso aos dados mesmo quando uma das réplicas não está disponível. O uso de réplicas também pode servir com mecanismo de redução de latência permitindo aos utilizadores acederem a réplicas geograficamente mais próximas. Com a replicação dos dados nasce a necessidade da manutenção e consistência das réplicas, actualizando-as regularmente de modo a que todos os dados replicados estejam consistentes entre si. Para esta consistência podem ser usados modelos que podem ser Optimistas ou Pessimistas.

Replicação Pessimista As técnicas de Replicação Pessimista[22] adoptam um modelo de consistência equivalente ao que se dispõe quando existe uma única cópia. Quando ocorre uma actualização, a réplica é obrigada a propagar as alterações a todas as outras, e só pode prosseguir quando receber uma confirmação das outras réplicas. Estes tipos de modelos chamam-se pessimistas por não permitirem concorrência de operações entre as réplicas. Este modelo assume que qualquer tipo de concorrência pode originar conflitos, ou seja, incoerência dos dados, então, não permite que haja concorrência.

Este tipo de técnicas são pouco escaláveis e não se adequam aos jogos multijogador. Uma réplica para actualizar o seu estado precisa da confirmação das outras o que implica uma grande troca de mensagens. Por outro lado, se uma das réplicas não estiver disponível então todo o sistema deixa de funcionar.

Replicação Optimista O que distingue as técnicas de Replicação Optimista[22] das pessimistas é o controlo de concorrência. Os algoritmos optimistas permitem que os dados existentes em réplicas sejam acedidos sem que estas se sincronizem com outras réplicas. Neste tipo de soluções assume-se que raramente vão ocorrer problemas que possam originar conflitos. Os conflitos ocorrem quando duas

réplicas do mesmo objecto são alteradas mas ainda não propagaram as alterações. As alterações das réplicas são feitas localmente e mais tarde são propagadas para as outras réplicas. Os conflitos são resolvidos quando são detectados e podem ser resolvidos de forma automática ou manual.

Esta solução possibilita uma maior disponibilidade e diminui o número de mensagens na rede. A Replicação Optimista é muito usada em jogos multi-jogador. Neste tipo de jogos é necessário sincronizar o estado das réplicas existentes nos vários clientes mantendo o estado consistente. O ideal é encontrar um meio-termo entre manter o jogo altamente consistente, e reduzir o número de mensagens necessárias. Um jogador normal tolera alguma inconsistência no jogo desde que isto não afecte a sua jogabilidade[17]. É a partir desta tolerância que se pode aplicar a redução de mensagens.

Limitação da Divergência As técnicas de Limitação da Divergência[13, 27, 12, 20] são técnicas de replicação optimista que toleram que o estado das réplicas divirjam entre si, desde que não ultrapassem um certo limite estabelecido. Estas técnicas são aplicáveis a jogos multi-jogador e têm como objectivo reduzir o número de mensagens trocadas.

Os limites, normalmente, são temporais e de ordem. Limites temporais servem para limitar o intervalo de tempo que uma réplica pode estar sem se sincronizar. Por exemplo, se atribuirmos um limite de 3 segundos, então isto significa que, no máximo, esta réplica só está inconsistente durante 3 segundos, nunca ultrapassando este limite. Os limites de ordem referem-se ao número de actualizações que ainda não foram aplicadas à réplica. Por exemplo, uma réplica com um limite N é considerada válida até se detectar que tem N actualizações pendentes. Após isto a réplica é actualizada.

O TACT[27] é um sistema que aplica técnicas de Limitação de Divergência como limites temporais e de ordem. Trata-se de um middleware aplicado às bases de dados e que possui flexibilidade na especificação dos limites permitindo combinações dos mesmos. A principal desvantagem do TACT é não ter em conta a lógica associada ao jogo. No TACT não existe qualquer noção de mapa de jogo, jogador, ou entidades do jogo.

Recuperação de Estado O Trailing State Synchronization (TSS)[7] é um algoritmo que permite a recuperação do estado quando é detectada uma incoerência na actualização do estado. Este algoritmo mantém réplicas do estado actual do jogo (estado principal), mas com um atraso temporal entre elas. Quando é detectada uma incoerência, o conteúdo de uma das réplicas é copiada para o lugar do estado principal. Como no estado anterior não existe ainda incoerência devido ao atraso temporal, as actualizações são novamente executadas na ordem correcta. Esta solução tem como desvantagem o armazenamento de várias réplicas do estado actual do jogo.

2.4 Gestão de Interesse

Nos jogos multi-jogador em que o jogador é representado por um avatar no mundo virtual, existe uma noção de percepção de objectos. O mundo virtual do jogo pode ser representado pelo mapa e todas as entidades do jogo. As entidades do jogo podem ser armas, paredes, avatar do jogador, etc... Cada jogador mantém uma cópia local do mundo virtual, que necessita de manter consistente. Acções efectuadas por jogadores, ou outras entidades do jogo, têm que ser propagadas para os jogadores afectados. Uma solução simples para este problema seria todas as entidades do jogo enviarem a todos os jogadores as suas acções. Esta solução tem um grande problema, um jogador vai receber actualizações de entidades que podem não lhe interessar, o que pode causar um excesso de mensagens na rede, que num ambiente de dispositivos móveis não é tolerável.

A Gestão de Interesse tira partido da noção de interesse do avatar dentro do jogo. Um avatar só necessita de receber eventos de um pequeno conjunto de entidades relevantes para ele. Esta técnica aplica-se na perfeição em jogos multi-jogador para dispositivos móveis em redes ad-hoc. Sendo um mecanismo que tem como objectivo reduzir o número de eventos que um jogador precisa de receber, então conseguimos reduzir o consumo da bateria por parte da interface de rede, bem como reduzir a memória necessária para o armazenamento e processamento dos eventos.

Existem três técnicas principais na área de Gestão de Interesse: Aura de Interesse, Campo de Visão e Divisão em Regiões.

Aura de Interesse A aura de interesse[4, 19, 26] tem como base a noção de interesse de um avatar. Um avatar dentro de um jogo apenas se consegue aperceber de outras entidades do jogo se estiverem dentro de uma certa distância de raio R. Normalmente, entidades que estejam relativamente perto do jogador têm um alto grau de detalhe e são muito perceptíveis, por outro lado, entidades distantes são muito pouco perceptíveis. Uma aura pode ser vista como um círculo/esfera que rodeia o avatar.

Neste modelo de Gestão de Interesse a aplicação cliente do jogador subscreve eventos que ocorram dentro da aura do avatar. Existem variações deste modelo que usam a distância do objecto ao avatar para calcular a frequência de envio de eventos, ou seja, quanto mais perto estiver o objecto do avatar, mais frequentemente são enviados os eventos.

Campo de Visão A aplicação de técnicas como o campo de visão[4, 8] é muito útil para Gestão de Interesse. Existem duas ideias principais que tornam esta ideia relevante: a amplitude de visão e a existência de obstáculos.

Um jogador normalmente só tem uma percepção de 180º a partir da posição em que se encontra[3], ou seja, não consegue ver objectos que se encontram atrás dele. Com isto podemos concluir que objectos que estejam atrás do jogador, não necessitam de enviar eventos ao jogador, ou pelo menos não com tanta frequência. Porém, é necessário ter atenção, a objectos que estejam atrás do

jogador, mas relativamente próximos. Se não houver qualquer tipo de envio de eventos de objectos que estejam atrás do jogador, se o jogador se virar de repente pode não ver de imediato os objectos que ali se encontram, o que pode afectar negativamente a jogabilidade. O que normalmente se aplica é um pequeno raio em redor do avatar. O avatar passa a receber sempre eventos de objectos que estejam muito próximos mesmo que estes se encontrem atrás do si.

Outra componente importante do campo de visão é a existência de obstáculos. Um jogador dentro de um jogo não consegue visualizar objectos que estejam atrás de obstáculos, como por exemplo, paredes, muros ou árvores. Como o jogador não consegue ver os objectos, então tem menor interesse neles, reduzindo assim o número de eventos que precisa de receber e processar.

Existem análises de vários algoritmos de Gestão de Interesse[4] que usam o conceito de Campo de Visão. Nesta análise podemos ver que o uso de técnicas de Campo de Visão, na existência de obstáculos, permitem reduzir o número de eventos que um jogador recebe até um factor de 6 vezes. O uso de mecanismos de Ray Visibility seria ideal para detectar que objectos estão visíveis ou não. O problema destes mecanismos é o consumo de recursos, mais propriamente CPU para efectuar os cálculos necessários das intersecções. Esta desvantagem tem um peso enorme na escalabilidade de um jogo em termos dos seus recursos de CPU, o que se agrava se estivermos no contexto de dispositivos de capacidades limitadas de CPU e bateria como os dispositivos móveis. Uma solução eficiente passa pelo algoritmo de TilePathDistance que usa triangulação do espaço do jogo através de polígonos. Os contornos dos polígonos são formados a partir dos limites do mundo e dos obstáculos existentes. Este algoritmo implica um pré-processamento para a triangulação do mapa.

Divisão em Regiões Na divisão do mundo virtual em regiões, o mundo virtual é repartido em várias regiões que não se intersectam entre si. Neste tipo de soluções de Gestão de Interesse, a aplicação cliente do jogador subscreve todos os eventos que possam ocorrer dentro da região em que se encontra, ou seja, qualquer evento que ocorra numa entidade dentro da região do jogador é imediatamente propagada para ele.

A divisão em zonas é muito usada para efeitos de balanceamento de carga[15, 6] nos jogos do tipo Massively Multiplayer Online Role-Playing Game (MMORPG). Como este tipo de jogos têm um grande número de jogadores são necessários múltiplos servidores. Para distribuir melhor a carga dos servidores, são atribuídas regiões a cada um deles. Um jogador apenas pode estar numa destas regiões, sendo que o seu estado é gerido pelo servidor da região.

As Regiões são então divisões que podem ser visíveis ou invisíveis para o utilizador. Divisões visíveis são aquelas de que o jogador tem a percepção de onde acaba a região. Nestas divisões, a transição de uma região para outra é normalmente feita com o uso de objectos especiais dentro do jogo como, por exemplo, um portal, porta da sala ou um aeroporto. As divisões invisíveis são aquelas que são transparentes para o utilizador. Nestas divisões o utilizador não tem qualquer noção do fim da região e a transição do jogador entre regiões é feita

de uma maneira automática e transparente. A divisão e atribuição de zonas aos servidores pode ser feita de uma maneira estática ou dinâmica. Se a atribuição for estática o mapa do jogo é dividido antes da execução e cada servidor tem um conjunto fixo de regiões. Após feita a atribuição já não é possível alterá-la. A atribuição dinâmica significa que os servidores podem alterar as dimensões das regiões que gerem e redistribuí-las por outros servidores. Por exemplo, um servidor que contenha um grande número de jogadores dentro da sua região, pode dividir a sua região ao meio, e atribuir essa metade a outro servidor que não esteja sobrecarregado.

Um exemplo de divisão dinâmica é a divisão dinâmica do mundo com base numa estrutura chamada N-Tree[26]. Cada região pode ser recursivamente dividida até um dado limite. As sub-regiões que resultam desta divisão são divididas pelos servidores.

Estando o mundo virtual partido em várias regiões, um jogador pode circular entre elas. Um servidor que gere uma região, quando detecta que um jogador está prestes a sair da região, tem que informar a aplicação cliente do jogador e informar o servidor para o qual o jogador se dirige. É necessário informar a aplicação cliente do jogador porque o seu servidor está prestes a mudar. Após mudar de região o estado do jogador passa a ser gerido por um novo servidor. O servidor antigo tem que informar o servidor novo do seu novo jogador, transferindo todo o estado que contêm acerca do jogador.

O tamanho das partições é um factor de grande peso neste tipo de soluções. Uma região grande tem como desvantagem o elevado número de eventos que um jogador pode receber mas reduz a frequência de troca de servidor associado ao jogador. Por outro lado, se uma região for pequena, um jogador recebe um número menor de eventos mas, em contrapartida, a transferência do jogador entre servidores ocorre mais frequentemente.

Aura de Interesse com Regiões Os mecanismos de Aura de Interesse e Divisão em Regiões podem ser aplicados em conjunto formando um mecanismo mais poderoso de Gestão de Interesse. Porém acrescenta um novo problema que não existia anteriormente, a proximidade de uma fronteira.

Como um jogador tem que receber actualizações de entidades que estejam dentro da sua aura, se um jogador se deslocar para perto de uma fronteira então vai necessitar de receber actualizações de entidades que estejam do outro lado da fronteira e que intersectem a sua aura. A troca de eventos de jogadores próximos das fronteiras mas em regiões diferentes, é um dos maiores problemas destas soluções e existem várias alternativas para resolver este problema.

Uma solução possível passa pelo lock de regiões e de objectos [2]. Um servidor a executar um evento que afecta uma área do mapa, como por exemplo uma explosão, tem que pedir o lock para essa área. Todos os eventos são propagados usando mecanismos de Publish/Subscribe.

Outra das soluções é a criação de uma região de subscrição à distância D da fronteira[5]. Todos os objectos que estejam a uma distância menor que D da fronteira, vão receber actualizações do outro lado da fronteira.

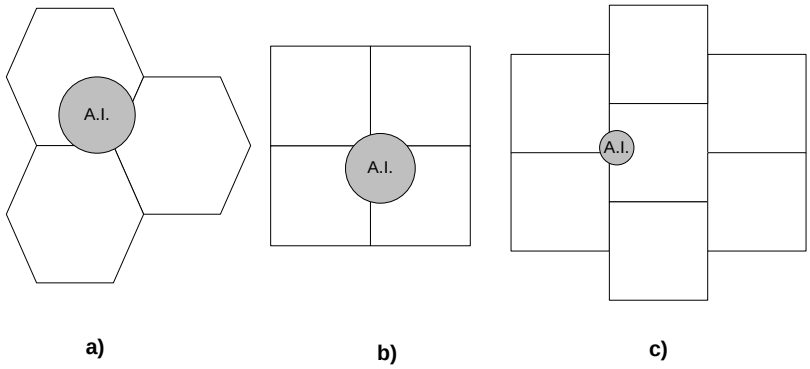

Fig. 1. Esta figura representa 3 tipos de partições de regiões. a) Partição em Quadrados b) Partição em Hexágonos c) Partição do tipo Brickworks

A forma que as regiões têm é um factor de peso quando se aplica uma aura em conjunto com divisão em regiões. Este problema é fácil de perceber analisando a Fig. 1. Podemos ver que se forem utilizadas regiões com a forma de quadrados, a aura do jogador intersecta até três regiões, o que significa que seria necessário comunicar com três servidores para haver troca de eventos. A forma de um hexágono é mais apropriada porque apenas é possível, no máximo, intersectar duas regiões, porém, o cálculo de distâncias usando hexágonos é mais complexa quando comparado com quadrados.

O Brickworks [19] é um tipo de partição que usa a simplicidade do cálculo de quadrados, com a vantagem do mesmo número de intersecções que a forma de um hexágono. Através da divisão das regiões de uma maneira intervalada, é possível obter os mesmos resultados da forma dum hexágono. A única condição desta solução é que o diâmetro da aura seja menor que metade do lado quadrado.

2.5 Sistemas Académicos - Gestão de Interesse

Nesta secção vão ser apresentados os sistemas académicos que mais se enquadram em jogos multi-jogador. Estes sistemas aplicam técnicas de Gestão de Interesse como Aura de Interesse e Campo de Visão.

Donnybrook O Donnybrook[16] é um sistema para jogos multi-jogador aplicados em arquitecturas P2P e que aplica várias técnicas de Gestão de Interesse. O Donnybrook tem como base 3 princípios: 1) Jogadores têm uma atenção limitada 2) Interacções importantes têm que ser feitas num curto intervalo de tempo e têm que ser consistentes 3) Realismo não deve ser afectado pela precisão.

O primeiro princípio tem como base o número de entidades nas quais uma pessoa se consegue focar. Um avatar dentro de um jogo só se consegue focar num dado conjunto de entidades, o que significa que o interesse de um jogador é

normalmente maior para certas entidades e menor para outras. As 3 regras que definem o interesse de um jogador numa entidade são a sua proximidade, mira do jogador e interacções recentes que tenham ocorrido.

O segundo princípio tem em conta interacções importantes e urgentes que ocorram entre dois jogadores. Supondo um cenário em que dois jogadores estão a tentar eliminar-se num jogo, é necessário que as interacções sejam o mais rápidas e consistentes possíveis, caso contrário a jogabilidade do jogo pode estar em risco. O Donnybrook usa um mecanismo de Entendimento Rápido entre Nós usando chamadas assíncronas entre apenas os jogadores envolvidos nessa interacção. Como usa um canal diferente dos canais usados para a transmissão de eventos, a sua transmissão é mais rápida.

O último princípio foca-se na lógica do jogo. Entidades que não estejam dentro do foco do jogador vão enviar os seus eventos menos frequentemente. Eventos menos frequentes podem resultar em movimentações bruscas de entidades do jogo que podem violar a lógica do jogo. Um exemplo disto são jogadores que estão numa posição e repentinamente aparecem noutro sítio. O Donnybrook aplica um mecanismo para guiar as entidades do jogo baseado em inteligência artificial.

Este sistema tem como principal desvantagem a dependência da lógica do jogo, estando orientado maioritariamente para jogos do tipo First Person Shooter (FPS). Uma das regras que define o interesse do jogador é o facto de a mira estar sobre o objecto. O conceito de mira deixa de fazer sentido quando aplicado em jogos que não sejam FPS.

Ring O Ring[8] é um sistema para ambientes virtuais de vários utilizadores que pode ser aplicado a jogos multi-jogador. Este sistema aplica mecanismos de Gestão de Interesse pois tem como objectivo a redução de mensagens que um jogador tem que receber para manter o seu estado consistente. Esta redução das mensagens vem da oclusão existente nos mapas dos jogos multi-jogador. Um jogador não consegue ver entidades do jogo que estejam escondidos atrás de paredes ou outro tipo de objectos e, como tal, não precisa de saber da existência de entidades que não consegue visualizar, evitando assim mensagens que, do ponto de vista do jogador, não vão trazer qualquer benesse.

Os jogadores mantêm uma cópia do estado do jogo que pode conter réplicas de objectos. Estas são réplicas de objectos que estão no campo de visão de jogador. A comunicação entre os jogadores é feita exclusivamente através dos servidores Ring, não havendo qualquer troca de mensagem entre os jogadores. Os servidores Ring servem então como filtro de mensagens, sabendo quais dos jogadores são visíveis a partir de uma dada localização, reencaminhado os eventos só para os jogadores que se conseguem aperceber deles.

Para a determinação da visibilidade que um jogador tem, é feita uma pré-computação ao mapa do jogo dividindo-o em células. Para cada célula são calculadas as células que são visíveis a partir dela, tendo em conta a existência dos obstáculos e são gravados esses resultados. Durante um jogo, para saber se um

jogador tem visibilidade para uma dada entidade, basta apenas determinar se essa entidade está numa das células visíveis calculadas anteriormente.

O Ring aplica um mecanismo de Gestão de Interesse com base no campo de visão do jogador que permite reduzir a largura de banda necessária dos clientes. O Ring apenas tem em conta se um jogador é visível ou não a partir de uma dada célula não tendo em conta outros factores que podem influenciar o interesse do jogador como, por exemplo, a proximidade. Neste sistema o tráfego gerado não está dependente do número de jogadores total, mas sim do número de jogadores cujos campos visuais se intersectem. A eficiência desta solução depende do conjunto de obstáculos existente no mapa. A obrigatoriedade do tráfego passar pelos servidores introduz uma maior latência, que pode aumentar consoante o número de servidores.

A3 O A3[3] é um algoritmo de gestão de Interesse, destinado a jogos do tipo MMORPG. Este algoritmo aplica os conceitos de proximidade, campo de visão e distância crítica. O interesse do jogador numa dada entidade é atribuído com base numa medida de relevância que varia entre 0 e 1. Uma relevância de 0 significa que o jogador não tem qualquer interesse na entidade não havendo qualquer envio de eventos. Por outro lado, uma relevância de 1 significa que a entidade é de extrema importância para o jogador originando uma grande frequência no envio de eventos.

A proximidade entre um jogador e uma dada entidade indica a relevância da entidade para o jogador. Quanto mais próximo da entidade, maior é a relevância da entidade o que implica maior frequência no envio de eventos. Outro dos conceitos aplicados no A3 é o conceito de campo de visão. Um jogador no mundo virtual tem interesse apenas em entidades que pode ver. O jogador não precisa de receber eventos de entidades que estejam atrás dele, fazendo com que a sua relevância seja menor.

Se um jogador se virar de repente, usando apenas o mecanismo acima referido, a entidade pode levar algum tempo a ser perceptível para o jogador. Para evitar este problema o A3 introduz o conceito de distância crítica. A distância crítica trata-se de um pequeno círculo que rodeia o avatar e que indica que, para o jogador, qualquer entidade que esteja dentro do círculo tem relevância máxima. Assim o jogador já tem noção das entidades que possam estar atrás dele, fazendo com que entidades próximas apareçam de imediato quando ele se vira.

Uma desvantagem deste algoritmo é a visão global de aura. Todos os jogadores têm o mesmo tipo de aura e são aplicadas as mesmas métricas de distâncias. Nos jogos podem haver jogadores com atributos especiais que podem, por exemplo, ver outros jogadores que estejam a uma longa distância, o que ia implicar outro tipo de aura para estes jogadores. Este algoritmo não é flexível nesse sentido pois aplica a mesma aura a todos os jogadores apenas possibilitando uma aura global.

Vector Field Consistency O Vector Field Concistency (VFC)[23] é um algoritmo de replicação Optimista baseado em Gestão de Interesse. Este Algoritmo

usa o conceito de Aura de Interesse. Quando aplicado a jogos multi-jogador o VFC permite reduzir o número de mensagens necessárias para manter as réplicas dos jogadores o mais consistente possível. As réplicas são cópias locais de objectos do jogo que cada jogador mantém. O VFC tem como principais conceitos Anéis de Consistência e Graus de Consistência.

Os Anéis de Consistência são anéis que se formam em redor de uma entidade chamada pivot. Um pivot é um objecto especial que define o grau de consistência de todos os objectos que o rodeia. Um bom exemplo de pivot é o avatar do jogador. Cada anel formado em redor do pivot tem um grau de consistência diferente. Tal como no modelo de Aura de Interesse, objectos mais próximos do pivot têm um grau de consistência maior enquanto que objectos mais afastados têm um grau de consistência menor.

No VFC um Grau de Consistência é especificado num vector tridimensional. As 3 dimensões do vector são: Temporal, Sequencial e de Valor. A dimensão temporal define o tempo máximo que uma réplica pode permanecer válida sem receber novas actualizações. A dimensão Sequencial define o número de actualizações que uma réplica pode não aplicar e permanecer operacional. A dimensão de Valor indica a diferença máxima entre uma réplica e o objecto original.

Os principais pontos fortes do VFC são a sua facilidade de percepção do seu modelo de consistência baseado em pivots e a flexibilidade na especificação de parâmetros do algoritmo como as 3 dimensões de um grau de consistência.

2.6 Dispositivos Móveis e Redes ad-hoc

As soluções actuais para jogos muti-jogador em ambientes ad-hoc para dispositivos móveis não têm em conta mecanismos de Gestão de Interesse que são indispensáveis para a redução de tráfego na rede.

Uma solução de partição do jogo[14] permite uma distribuição de carga entre o servidor do jogo e os clientes. Quando o servidor começa a ficar com poucos recursos disponíveis como, por exemplo, memória, CPU ou largura de banda, o jogo é repartido, distribuindo funções do servidor pelos clientes existentes, aliviando assim a carga do servidor. Esta divisão é feita tendo em conta a disponibilidade de memória, capacidade de processamento e largura de banda disponível nos clientes. A divisão tem que ser feita de uma maneira optimizada pois a partição da aplicação pode originar uma maior quantidade de tráfego na rede devido à distribuição das funções por vários clientes.

Uma rede ad-hoc é uma rede heterogénea que pode conter vários dispositivos de níveis de performance diferentes como telemóveis, tablets ou portáteis. Os dispositivos comunicam através de canais de informação sem fios, cooperando entre eles para o reencaminhamento de pacotes. As duas arquitecturas mais usadas para jogos em redes ad-hoc são: Cliente-Servidor e Peer-to-Peer (P2P).

Na arquitectura Cliente-Servidor o dispositivo que corresponde ao servidor tem a responsabilidade de gerir a lógica do jogo e actua como um coordenador da rede. Os clientes enviam os dados para o servidor, que processa a informação, e envia respostas aos clientes afectados. Esta é uma arquitectura simples de implementar e tem como principal desvantagem a existência de apenas um servidor.

Como apenas existe um servidor, se este se desligar, toda a rede fica parada enquanto o servidor não se ligar ou for eleito um novo servidor. A escalabilidade é outra das desvantagens que a arquitectura apresenta. O servidor tem que lidar com todo o tráfego da rede, podendo tornar-se o *bottleneck* da rede.

Numa arquitectura P2P o dispositivo de cada jogador mantém uma cópia local do jogo e informa os dispositivos dos outros jogadores quando ocorrem actualizações. Esta arquitectura é mais difícil de implementar pois cada cliente tem que lidar com a lógica do jogo, não havendo uma entidade única para o efeito. Apesar desta desvantagem, uma arquitectura P2P tem uma maior tolerância a faltas que uma arquitectura Cliente-Servidor e uma maior escalabilidade. A paragem de um dispositivo não leva à paragem do sistema e não existe um ponto único na rede para onde todo o tráfego converge.

Existe uma solução a nível de arquitectura de redes ad-hoc baseada em Zone Servers [21]. Esta solução tenta juntar as vantagens de uma arquitectura Cliente-Servidor com uma arquitectura Peer-to-Peer. Nesta arquitectura existem 2 tipos de nós na rede: Nós normais e ZoneServers. Os ZoneServers são clientes que são eleitos com base nos recursos que têm disponíveis como, por exemplo, CPU, memória, largura de banda ou bateria. Cada ZoneServer gere um pequeno grupo de jogadores, recebendo os updates dos seus jogadores e propago-os para outros jogadores através dos outros ZoneServers existentes na rede. Se um destes ZoneServers deixar de responder, os jogadores associados a ele podem-se ligar a outros servidores e continuar a jogar. A descoberta dos ZoneServers é feita com base no protocolo Service Location Protocol (SLP) em que os clientes fazem multicast de um URL do jogo e os servidores respondem. A principal desvantagem desta arquitectura é funcionar apenas ao nível da rede não usando qualquer mecanismo de Gestão de Interesse.

Os sistemas apresentados aplicam técnicas Gestão de Interesse de jogos multi-jogador mas não estão focados para as limitações que os telemóveis apresentam como a autonomia limitada da bateria e a fraca capacidade de processamento. Os sistemas mostram ter pouca flexibilidade e alguns deles estão focados para um tipo de jogo específico, não permitindo a aplicação do sistema em jogos de outro tipo.

3 Solução

A solução deste trabalho consiste na implementação do modelo de consistência VFC na plataforma Android. O sistema é um middleware que irá lidar com a consistência dos jogos multi-jogador através da aplicação do VFC. O VFC terá então o papel de filtro durante a execução, que selecciona quais as actualizações que devem ser ou não enviadas. Na secção 3.1 encontra-se uma descrição detalhada do VFC e na secção 3.2 está a descrição da arquitectura do nosso middleware.

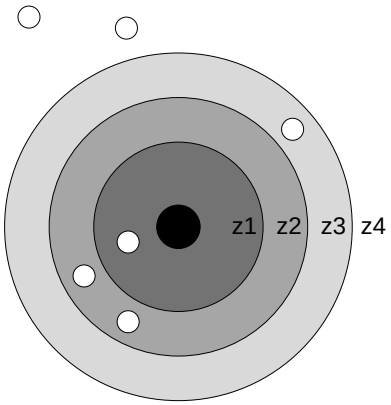

Fig. 2. Anéis de Consistência do VFC: Pivot representado por ponto preto e objectos normais representados por pontos brancos. Os anéis são representados por z[i].

3.1 Algortimo - VFC Vector Field Consistency

O VFC Trata-se de um algoritmo de replicação Optimista e que aplica conceitos de Limitação de Divergência. Este algoritmo ajusta dinamicamente a consistência dos objectos replicados, com base no estado actual do jogo, gerindo o grau de consistência de cada objecto com base na sua distância a objectos especiais. Com o VFC consegue-se reduzir o número de mensagens que é preciso trocar entre jogadores em jogos multi-jogador. Esta redução de mensagens vai corresponder a um menor consumo de energia, memória e processamento dos dispositivos móveis.

O VFC tem como principais conceitos: Anéis de Consistência e Graus de Consistência.

Anéis de Consistência No VFC cada jogador tem uma vista local do mundo virtual correspondente ao jogo. Dentro de uma vista existem objectos especiais chamados pivots que definem o grau de consistência de todos os objectos que os rodeiam. O grau de consistência de um dado objecto é dado pela função da distância do objecto ao pivot mais próximo. Isto significa que objectos próximos terão um grau de consistência mais alto enquanto que objectos mais distantes terão um grau de consistência mais baixo. Num jogo um objecto pivot pode ser por exemplo o avatar que representa o jogador.

Os anéis de consistência são anéis formados em redor dos objectos pivots e que definem o grau de consistência dos objectos que se situam nesses anéis. Na Fig. 2 temos um exemplo de 3 anéis de consistência em redor de um objecto pivot. Neste desenho o ponto preto representa o objecto pivot e os pontos brancos representam os objectos existentes na vista do jogador. A intensidade da cor existente nos

anéis representa o grau de consistência desse anel. Neste caso o objecto situado no anel z1 terá um grau de consistência maior que os dois objectos situados no anel z2 e assim sucessivamente. O z4 representa a zona que está fora dos anéis de consistência. O grau de consistência de z4 pode ter um valor ou pode ser nulo. A nulidade do grau de consistência na zona z4, está dependente da lógica do jogo. Para este exemplo apenas se consideraram 3 anéis mas a flexibilidade do VFC permite-nos definir N anéis de consistência.

Graus de Consistência Cada anel de consistência tem um grau de consistência associado a ele. Um grau de consistência é um vector de 3 dimensões que especifica a divergência máxima permitida para objectos dentro desse anel. As 3 dimensões existentes neste vector são dimensões de Temporal(θ), Sequencial(σ) e Valor(υ).

A Dimensão Temporal θ especifica o intervalo máximo temporal de dessincronização de duas réplicas. Nesta dimensão especificamos o intervalo máximo de tempo (segundos) que uma réplica de um objecto pode ficar sem receber actualizações. Por exemplo se escolhermos um intervalo de θ =2 segundos significa que no máximo a réplica está desactualizada 2 segundos.

A Dimensão Sequencial σ especifica o número máximo de actualizações que uma réplica pode perder. Com esta dimensão garantimos que uma réplica de um objecto está no máximo desactualizada em σ actualizações face ao objecto original. Considerando como exemplo que $\sigma = 2$ e o estado do objecto original e das suas réplicas estão sincronizados, então o objecto original pode aplicar até 2 actualizações e só à 3ª actualização é que propaga as actualizações para as suas réplicas.

A Dimensão de Valor υ especifica a diferença máxima de estado permitida entre a réplica e o objecto original. Esta dimensão necessita de uma função que calcule a percentagem de diferença entre 2 objectos o que a torna dependente da implementação do jogo. Se por exemplo usarmos um valor $\upsilon = 25\%$ significa que o estado entre o objecto original e a sua réplica tem uma diferença máxima de 25%.

Considerando agora como exemplo um grau de consistência representado pelo vector $[\ \theta\ ;\ \sigma\ ;\ \upsilon\ ] = [2\ ;\ 3\ ;\ 30]$ podemos concluir que os objectos existentes no anel com esse vector, estão desactualizados no máximo 2 segundos ou podem perder até 3 actualizações ou podem diferir de estado em relação ao objecto original em 30%. A partir do momento que o valor de qualquer uma das dimensões é atingido ou ultrapassado, dá-se inicio à actualização da réplica.

Generalizações do VFC O VFC permite a aplicação de algumas generalizações: multi-pivot e multi-anéis. Multi-pivot significa que podem existir vários pivots na mesma vista. Com a existência de vários pivots um objecto pode ter vários graus de consistência diferentes sendo cada grau atribuído por um pivot. Neste caso o grau de consistência escolhido é o maior dos existentes.

A outra generalização, multi-aneis, permite-nos atribuir graus e anéis de consistência diferentes para objectos diferentes. Esta generalização permite que um

objecto à mesma distância do pivot que outro objecto, possa ter um grau de consistência diferente do outro objecto. Por exemplo, num jogo existem objectos mais prioritários que outros. Se um jogador estiver próximo de outro então provavelmente a consistência entre eles terá que ser alta mas se considerarmos outro objecto à mesma distância como um pack de munições então a consistência será menor.

Especificação do Modelo de Consistência O Modelo de consistência do VFC é o conjunto de todos os objectos, pivots, anéis de consistência e graus de consistência. Do ponto de vista do programador para especificar todo o modelo de consistência associado a qualquer jogo multi-jogador basta apenas especificar estes 4 conjuntos. A agregação destes 4 conjuntos corresponde ao parâmetro ϕ do VFC.

Vantagens do VFC Este algoritmo tem como principais vantagens a facilidade de programação programador, manter a jogabilidade do jogo aceitável e o aumento na eficiência no uso dos recursos disponíveis.

A facilidade de especificação tem origem na fácil percepção por parte do programador. O modelo de consistência baseado em pivots é intuitivo e fácil de perceber. A flexibilidade e simplicidade do VFC na definição dos graus de consistência tornam o algoritmo aplicável a vários tipos de jogos. Ao contrário de outros sistemas, o VFC é bastante flexível na definição de diferentes tipos de consistência para objectos diferentes.

A jogabilidade do jogador é pouco afectada com este algoritmo. Do ponto de vista do jogador todas as regras do jogo são cumpridas. Isto acontece porque a informação relevante na lógica do jogo é, propagada o mais rapidamente possível para os clientes interessados enquanto a que informação menos relevante pode ser atrasada não afectando a lógica do jogo.

Uma maior eficiência no uso dos recursos disponíveis é a principal vantagem do uso do VFC especialmente em dispositivos móveis. Tratando-se de um algoritmo de Gestão de Interesse apenas propaga actualizações para os jogadores interessados, reduzindo assim o número de mensagens necessárias para a execução do jogo. Esta selecção e diminuição de mensagens enviadas permite reduzir a largura de banda necessária para a execução do jogo e permite mascarar a latência existente na rede através da atribuição de prioridades consoante o grau de consistência, ou seja, as mensagens mais relevantes são enviadas de imediato, enquanto que as mensagens menos relevantes são atrasadas.

Com a aplicação do VFC aumentamos a autonomia dos dispositivos móveis de várias maneiras. A redução de mensagens tem como impacto uma redução no uso de interfaces de rede. Estas interfaces em dispositivos móveis têm um grande impacto na autonomia da bateria pois gastam muita energia. Por outro lado, a redução no número de mensagens diminui a carga de processamento do dispositivo.

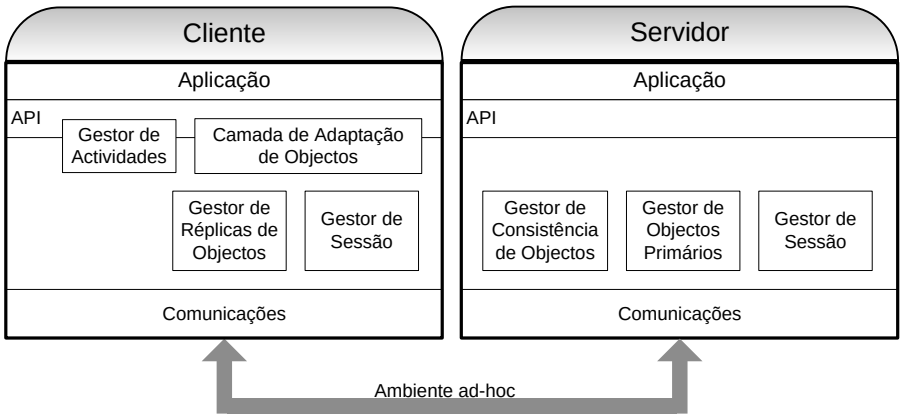

Fig. 3. Representação da arquitectura do sistema.

3.2 Arquitectura

A arquitectura do sistema é baseada numa arquitectura Cliente-Servidor apresentada na Fig. 3 e irá servir como um middleware para a aplicação/jogo. A base da escolha Cliente-Servidor veio do protocolo usado nas comunicações entre os nós na rede, o Bluetooth. Por limitação da tecnologia Bluetooth todas a mensagens geradas têm que passar obrigatoriamente por um nó especial da rede, que no nosso caso será o nó que representa o Servidor. Apesar disto, a escolha do Bluetooth vem essencialmente do baixo consumo de energia que esta tecnologia apresenta na troca de mensagens.

O protocolo Cliente-Servidor é implementado na componente de Gestor de Sessão. Do lado do servidor, esta componente tem as responsabilidades de gerir os locks de escrita e propagar as actualizações pelos outros nós. Toda a parte da comunicação entre clientes e servidor será implementada na Camada de Comunicações seguindo uma topologia do tipo estrela.

Leitura e Escrita de Objectos Todo o estado do jogo é representado em objectos de 2 tipos: Réplicas e Objectos Primários. As Réplicas são as cópias dos Objectos Primários, e são armazenadas localmente no lado dos clientes. A componente responsável pelas Réplicas é o Gestor de Réplicas de Objectos e a componente responsável pelos objectos Primários é o Gestor de Objectos Primários. A Camada de Adaptação de objectos tem a função de converter a representação da informação usada pela aplicação do cliente, para a representação interna do nosso sistema.

A aplicação do lado do cliente efectua todas as leituras que necessita a partir das Réplicas localmente. O facto de o cliente ler apenas os dados armazenados localmente, sem ter que consultar o servidor, aumenta significativamente o tempo de leitura da informação. Como o cliente pode ler informação desactualizada este comportamento segue o modelo de Replicação Optimista.

Ao contrário das leituras que são feitas localmente, para a aplicação do cliente efectuar operações de escrita, necessita do consentimento do servidor. Este consentimento é feito através de um mecanismo de locks gerido centralmente pelo Gestor de Sessão do Servidor. Este mecanismo é necessário para lidar com a concorrência de escritas de vários clientes. Após o Servidor ceder o lock ao cliente, são enviadas as actualizações, seguidas de uma mensagem a informar que os locks não são mais necessários. Concluído este processo de escrita, dá-se inicio á propagação das actualizações pelos outros clientes, conforme a especificação dos parâmetros do VFC.

Propagação de Actualizações Para a propagação das actualizações pelos clientes, o Gestor de Sessão do Servidor aplica um conceito de Ronda. Basicamente, o Gestor de Sessão executa periodicamente uma função, com o objectivo de informar os clientes necessários sobre as novas actualizações.

Durante cada ronda, o Servidor calcula as actualizações que é necessário enviar para cada cliente, com base na especificação dos parâmetros do VFC. Após o cálculo, é enviada uma Mensagem de Ronda com todas actualizações. Quando o cliente recebe as actualizações, aplica as actualizações às suas Réplicas através do Gestor de Réplicas de Objectos.

A recepção periódica de actualizações por parte do cliente permite sincronizar a aplicação do cliente com o sistema, através de callback's implementados pelo Gestor de Actividades. O Gestor de actividades permite informar a aplicação que o estado do jogo se alterou, permitindo, por exemplo, à aplicação actualizar a informação apresentada no ecrã.

Aplicação do VFC O Gestor de Consistência de Objectos é a única componente responsável pela aplicação do VFC. Esta componente é usada pelo Gestor de Sessão durante as Rondas e pedidos de escrita dos clientes. O funcionamento do Gestor de Sessão está dividido em duas fases, a fase de configuração e fase activa. Durante a fase de configuração os clientes registam os objectos que devem ser partilhados e enviam os seus parâmetros de consistência (ϕ do VFC). A fase activa é a fase de funcionamento normal, em que o Servidor processa todos os pedidos de escrita dos clientes, e executa a função periódica de Rondas.

Arquitectura do Android O Android[1] é uma plataforma para dispositivos móveis, baseada na linguagem java, que possuí uma máquina virtual própria chamada Dalvik Virtual Machine (DVM). Ao contrário da Java Virtual Machine, a DVM foi construída focando-se nas limitações dos dispositivos móveis como a memória, CPU e bateria.

Existem 2 aspectos relevantes da arquitectura do Android relativos a este trabalho: Threads e Comunicação.

O Android possui suporte para threads bem como mecanismos de comunicação entre elas como *Handlers* e *AsyncTasks*. As threads são essenciais em

[1] Página oficial: `http://developer.android.com`

jogos, para não influenciar a jogabilidade do jogo as operações mais pesadas devem ser feitas em background usando threads separadas.

Relativamente à comunicação, o Android não permite o uso do java RMI. Para a comunicação entre dispositivos o Android disponibiliza comunicações com Sockets. Estes sockets podem usar tecnologias como o WiFi, Bluetooth ou GPRS. O suporte de Serialização de objectos é também um mecanismo importante para a transmissão de objectos.

4 Metodologia de Avaliação

Para a avaliação deste trabalho será desenvolvido um jogo demonstrativo das vantagens do VFC. O jogo desenvolvido é um jogo 2D baseado no clássico *Asteroids* em que o objectivo do jogador é não deixar a sua nave ser abatida pelos asteróides.

Serão usadas 2 tipos de análises neste trabalho: Quantitativas e Qualitativas.

O objectivo da análise quantitativa é medir o tráfego gerado entre vários jogadores tendo em conta 2 tipos soluções. Na primeira solução vamos usar o VFC com consistência máxima. Isto significa que vamos aplicar o VFC com difusão de todas as mensagens para todos os jogadores. A segunda solução será usar o VFC o mais optimizado possível, especificando os parâmetros ϕ do algoritmo. Com esta análise vamos mostrar uma redução no tráfego gerado através do uso do VFC.

A análise qualitativa tem como objectivo medir a jogabilidade do jogo implementado, ou seja, mostrar que o uso do VFC não afecta a jogabilidade do ponto de vista do jogador. Para isto vamos pedir a várias pessoas para testarem o jogo e darem o seu feedback da jogabilidade do mesmo, quando usamos VFC consistência máxima VS VFC optimizado.

5 Conclusão

Neste trabalho apresentámos vários mecanismos que permitem mascarar a latência numa rede e reduzir o tráfego necessário entre jogos multi-jogador. A técnica mais usada na redução de tráfego em jogos multi-jogador, é a técnica de Gestão de Interesse, que permite seleccionar apenas os eventos que são relevantes para aquele jogador, descartando assim os menos relevantes. Relativamente aos dispositivos móveis e redes ad-hoc, é pouco o trabalho desenvolvido no âmbito da consistência de estado em jogos multi-jogador. Os sistemas actuais não estão optimizados para os dispositivos móveis e não são flexíveis quanto ao tipo de jogo e especificação do modelo de consistência.

A solução deste trabalho consiste na implementação do modelo de consistência VFC na plataforma Android. Para a avaliação será implementado um jogo e serão avaliados os principais aspectos de redução de mensagens necessárias entre os dispositivos móveis e impacto da jogabilidade com o uso da nossa solução.

References

1. Ieee standard for information technology- telecommunications and information exchange between systems- local and metropolitan area networks- specific requirements part ii: Wireless lan medium access control (mac) and physical layer (phy) specifications. *IEEE Std 802.11g-2003 (Amendment to IEEE Std 802.11, 1999 Edn. (Reaff 2003) as amended by IEEE Stds 802.11a-1999, 802.11b-1999, 802.11b-1999/Cor 1-2001, and 802.11d-2001)*, 2003.
2. Marios Assiotis and Velin Tzanov. A distributed architecture for mmorpg. In *Proceedings of 5th ACM SIGCOMM workshop on Network and system support for games*, NetGames '06, New York, NY, USA, 2006. ACM.
3. Carlos Eduardo Bezerra, Fábio R. Cecin, and Cláudio F. R. Geyer. A3: A novel interest management algorithm for distributed simulations of mmogs. In *Proceedings of the 2008 12th IEEE/ACM International Symposium on Distributed Simulation and Real-Time Applications*, DS-RT '08, pages 35–42, Washington, DC, USA, 2008. IEEE Computer Society.
4. Jean-Sébastien Boulanger, Jörg Kienzle, and Clark Verbrugge. Comparing interest management algorithms for massively multiplayer games. In *Proceedings of 5th ACM SIGCOMM workshop on Network and system support for games*, NetGames '06, New York, NY, USA, 2006. ACM.
5. Wentong Cai, Percival Xavier, Stephen J. Turner, and Bu-Sung Lee. A scalable architecture for supporting interactive games on the internet. In *Proceedings of the sixteenth workshop on Parallel and distributed simulation*, PADS '02, pages 60–67, Washington, DC, USA, 2002. IEEE Computer Society.
6. Jin Chen, Baohua Wu, Margaret Delap, Björn Knutsson, Honghui Lu, and Cristiana Amza. Locality aware dynamic load management for massively multiplayer games. In *Proceedings of the tenth ACM SIGPLAN symposium on Principles and practice of parallel programming*, PPoPP '05, pages 289–300, New York, NY, USA, 2005. ACM.
7. Eric Cronin, Burton Filstrup, Anthony R. Kurc, and Sugih Jamin. An efficient synchronization mechanism for mirrored game architectures. In *Proceedings of the 1st workshop on Network and system support for games*, NetGames '02, pages 67–73, New York, NY, USA, 2002. ACM.
8. Thomas A. Funkhouser. Ring: A client-server system for multi-user virtual environments. In *Symposium on Interactive 3D Graphics*, pages 85–92, 1995.
9. Carsten Griwodz. State replication for multiplayer games. In *Proceedings of the 1st workshop on Network and system support for games*, NetGames '02, pages 29–35, New York, NY, USA, 2002. ACM.
10. J. Haartsen. Bluetooth - the universal radio interface for ad-hoc, wireless connectivity. Technical report, 1998.
11. Andreas Janecek and Helmut Hlavacs. Programming interactive real-time games over wlan for pocket pcs with j2me and .net cf. In *Proceedings of 4th ACM SIGCOMM workshop on Network and system support for games*, NetGames '05, pages 1–8, New York, NY, USA, 2005. ACM.
12. Narayanan Krishnakumar and Arthur J. Bernstein. Bounded ignorance: a technique for increasing concurrency in a replicated system. *ACM Trans. Database Syst.*, 19:586–625, December 1994.
13. Narayanan Krishnakumar and Ravi Jain. Escrow techniques for mobile sales and inventory applications. *Wirel. Netw.*, 3:235–246, August 1997.

14. Madan Kumar M. M, Amit Thawani, Sridhar V, and Y. N. Srikant. Analysis of application partitioning for massively multiplayer mobile gaming. In *Proceedings of the 1st international conference on MOBILe Wireless MiddleWARE, Operating Systems, and Applications*, MOBILWARE '08, pages 28:1–28:6, ICST, Brussels, Belgium, Belgium, 2007. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).
15. Dugki Min, Donghoon Lee, Byungseok Park, and Eunmi Choi. A load balancing algorithm for a distributed multimedia game server architecture. In *Proceedings of the IEEE International Conference on Multimedia Computing and Systems - Volume 2*, ICMCS '99, pages 882–, Washington, DC, USA, 1999. IEEE Computer Society.
16. Jeffrey Pang. Scaling peer-to-peer games in low-bandwidth environments. In *In Proc. 6th Intl. Workshop on Peer-to-Peer Systems (IPTPS*, 2007.
17. Lothar Pantel and Lars C. Wolf. On the impact of delay on real-time multiplayer games. In *NOSSDAV '02: Proceedings of the 12th international workshop on Network and operating systems support for digital audio and video*, pages 23–29, New York, NY, USA, 2002. ACM.
18. Lothar Pantel and Lars C. Wolf. On the suitability of dead reckoning schemes for games. In *Proceedings of the 1st workshop on Network and system support for games*, NetGames '02, pages 79–84, New York, NY, USA, 2002. ACM.
19. K. Prasetya and Z. D. Wu. Performance analysis of game world partitioning methods for multiplayer mobile gaming. In *Proceedings of the 7th ACM SIGCOMM Workshop on Network and System Support for Games*, NetGames '08, pages 72–77, New York, NY, USA, 2008. ACM.
20. Nuno Preguiça, J. Legatheaux Martins, Miguel Cunha, and Henrique Domingos. Reservations for conflict avoidance in a mobile database system. In *Proceedings of the 1st international conference on Mobile systems, applications and services*, MobiSys '03, pages 43–56, New York, NY, USA, 2003. ACM.
21. Sebastián Matas Riera, Oliver Wellnitz, and Lars Wolf. A zone-based gaming architecture for ad-hoc networks. In *Proceedings of the 2nd workshop on Network and system support for games*, NetGames '03, pages 72–76, New York, NY, USA, 2003. ACM.
22. Yasushi Saito and Marc Shapiro. Optimistic replication. *ACM Comput. Surv.*, 37:42–81, March 2005.
23. Nuno Santos, Luís Veiga, and Paulo Ferreira. Vector-field consistency for ad-hoc gaming. In *Proceedings of the ACM/IFIP/USENIX 2007 International Conference on Middleware*, Middleware '07, pages 80–100, New York, NY, USA, 2007. Springer-Verlag New York, Inc.
24. Jouni Smed, Timo Kaukoranta, Harri Hakonen, and Oy L M Ericsson Ab. A review on networking and multiplayer computer games. Technical report, 2002.
25. Luís Veiga, André Negrão, Nuno Santos, and Paulo Ferreira. Unifying divergence bounding and locality awareness in replicated systems with vector-field consistency. *Journal of Internet Services and Applications*, 1:95–115, 2010. 10.1007/s13174-010-0011-x.
26. Shi Xiang-bin, Wang Yue, Li Qiang, Du Ling, and Liu Fang. An interest management mechanism based on n-tree. In *Proceedings of the 2008 Ninth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing*, pages 917–922, Washington, DC, USA, 2008. IEEE Computer Society.

27. Haifeng Yu and Amin Vahdat. Design and evaluation of a conit-based continuous consistency model for replicated services. *ACM Trans. Comput. Syst.*, 20:239–282, August 2002.