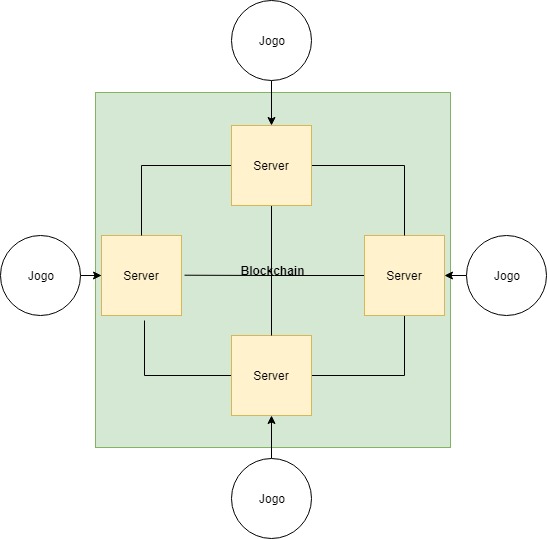
Descrição do Protocolo de Comunicação

Descrição Geral

Devido à característica cliente servidor da aplicação, foi implementado um protocolo de comunicação dessa aplicação. Este protocolo tem o *stop-and-wait* [SAW] como método para controle de fluxo. Foi construído sobre o protocolo TCP/IP. Este é dividido em duas partes principais. A primeira parte consiste na comunicação entre o processo que constitui o jogo em execução e o processo que funciona como um servidor local para esta instância do jogo. E a segunda parte é a comunicação entre a instância local de servidor com os demais processos servidores que constituem os nós da rede *Blockchain* Estes serão mais detalhadamente explicados na seção posterior e a visão global pode ser vista na Figura...



A próxima subseção descreve as mensagens do protocolo.

O protocolo

As mensagens são classificadas em três categorias. A primeira categoria é composta pelas mensagens de eventos que são utilizadas quando um jogador realiza algum tipo ação no jogo. A ação é requisitada junto ao servidor local e este envia para os demais servidores dos demais jogadores. As mensagens de controle que são usadas quando um novo jogador é adicionado ou um jogador já existente se conecta ao jogo e portanto, conecta-se a sua instância de servidor local. Por último, tem-se as mensagens de requisição e espalhamento de blocos, esta ocorre entre servidores: Quando um servidor se conecta a rede, este receberá os blocos desde que foram validados desde a última vez que se conectou; Quando um bloco é validado, neste caso o bloco validado é espalhado para os demais servidores conectados; Quando um novo jogador adentra o jogo e o seu servidor local deve se conectar a primeira vez com a Blockchain e portanto receberá toda a cadeia de blocos.

1. Comunicação jogo-servidor
2. Estabelecendo/Encerrando conexão

Sempre que uma instância cliente do jogo é criada, o novo usuário faz autenticação ao jogo, uma conexão com o servidor local deve ser estabelecida. O cliente comunica-se com o servidor local através de troca de mensagens. Portanto as mensagens de estabelecimento de conexão são realizadas quando a autenticação do usuário é realizada, e de finalização quando o usuário se desconecta do jogo. Seguem os seguintes padrões:

* LOGIN USER KEY - “LOGIN USER KEY\n”
  + Onde USER é uma string que representa o *login* que o usuário está cadastrado no jogo e KEY é outra string contendo o hash da senha relacionada aquele usuário de USER. Ambas informações são concatenadas a “LOGIN” e separadas por 1 espaço cada, finalizando a mensagem com um “\n”.
  + Ex: “LOGIN FULANO 123456\n”, “LOGIN CAIO 244466666\n”, etc.
* REGISTER JSON - “REGISTER JSON\n”
  + Onde JSON é uma string que representa o objeto Player que o usuário está cadastrado no jogo. O JSON é concatenado a “REGISTER” e separados por 1 espaço, finalizando a mensagem com um “\n”.
* LOGOUT - “LOGOUT\n”
  + A string “LOGOUT\n é enviada pelo servidor que deseja encerrar a conexão e não é esperada mensagem de confirmação.

1. Mensagem de confirmação/erro

Mensagens de confirmação são enviadas sempre depois de receber algum evento de jogador, login ou registro. Podendo também indicar fim de um bloco ou um eventual erro. Não esperam mensagem em retorno e seguem o padrão abaixo:

* OK - “OK\n”
  + Uma string “OK\n” que indica o recebimento correto da mensagem enviada anteriormente.
* END BLOCK - “END BLOCK\n”
  + String “END BLOCK\n” que indica o fim do envio de comandos de um bloco por parte do servidor.
* ERROR COD - “ERROR COD\n”
  + Onde COD representa um inteiro que está relacionado ao erro ocorrido. Sendo portanto concatenado na string “ERROR”, separados por um espaço e finaliza a mensagem com “\n”.
  + Ex.: “ERROR 0\n”, “ERROR 127\n”, “ERROR 7\n”, etc.
* NOT FOUND - “NOT FOUND\n”
  + Mensagem enviada em resposta a mensagem SAVED GAMESTATE, no caso em que nenhum *game state* salvo foi encontrado na Blockchain. É formada por uma string única “NOT FOUND” concatenada ao terminador “\n”.
  + Ex.: “NOT FOUND\n”

1. Eventos de Jogador

Toda ação realizada por um jogador interferirá no jogo de todos os demais, portanto estas devem ser mensagens enviadas ao servidor. Este deverá garantir que todos vejam as mudanças e que estas ocorram na mesma ordem. Todos os comandos são concatenados a uma string inicial “COMMAND ” antes de serem enviadas ao servidor. As ações possíveis por parte do jogador e a estrutura das suas mensagens seguem os padrões abaixo:

* ATTACK PLAYER ISLAND FROM MYISLAND - “ATTACK PLAYER ISLAND FROM MYISLAND\n”
  + Mensagem padrão de ataque, devem ser informados o destino (usuário que irá ser atacado). Onde PLAYER é uma string que representa o *login* que o usuário que está sendo atacado está cadastrado no jogo. ISLAND que representa o ID da ilha do player atacado. E MYISLAND é o ID da ilha atacante. A informação é concatenada a string “ATTACK” e string “FROM” e separadas por 1 espaço finalizando a mensagem com um “\n”. Uma mensagem de confirmação é aguardada.
  + Ex.: “COMMAND ATTACK CAIOVR 0 FROM 1\n”, “ COMMAND ATTACK FULANO 3 FROM 0\n”, etc.
* EXPLORE ISLAND ID - “EXPLORE ISLAND ID\n”
  + Mensagem de Exploração de ilha, uma nova ilha é gerada randomicamente e o exército da ilha que tem o ID especificado na mensagem é enviado para a exploração. ISLAND é uma string JSON que representa a ilha que será criada. ID é um inteiro que representa o id da ilha que o jogador que está fazendo a exploração possui e de onde o exército partirá para a exploração. O inteiro ID é concatenado a string JSON “ISLAND” e a string “EXPLORE”, separado por um espaço e finalizado com um “\n”. Uma mensagem de confirmação é aguardada.
  + Ex.: “COMMAND EXPLORE ISLAND 98\n”, “COMMAND EXPLORE ISLAND 42\n”, etc.
* CHANGE OLD\_ROLE QTD TO NEW\_ROLE IN MYISLAND - “CHANGE OLD\_ROLE QTD TO NEW\_ROLE IN MYISLAND\n”
  + Mensagem que tem função de trocar as classes dos cidadãos da ilha. Onde OLD\_ROLE e NEW\_ROLE são strings que podem assumir os seguintes valores: “NOT DEFINED”, “TROOPER”, “RANGED”, “WORKER\_WOOD”, “WORKER\_STONE”, “WORKER\_GOLD”, “WORKER\_FOOD”. E QTD um valor inteiro positivo maior que 0. Estas são concatenadas a string “CHANGE” e “TO”, separados por um espaço cada e finalizando a mensagem com “\n”. Aguarda uma mensagem de confirmação do servidor.
  + Ex.: ”CHANGE WORKER\_WOOD 30 TO TROOPER\n”, “CHANGE RANGED 2 TO TROOPER\n”, “CHANGE TROOPER 20 TO WORKER\_GOLD\n”, etc.
* CREATE ARMY TYPE QTD - “CREATE ARMY TYPE QTD\n”.
  + Mensagem para a criação de exército na ilha. TYPE é um tipo string que representa o tipo de exército que será criado, podendo ser do tipo: SOLDADO,e QTD é um inteiro que estabelece a quantidade que será criada. Ambos são concatenados a string “CREATE ARMY” e separados por um caractere de espaço vazio entre eles, finalizando a mensagem com o caractere “\n”. Uma mensagem de confirmação vinda do servidor é aguardada.
  + Ex.: “COMMAND CREATE ARMY SOLDADO 1\n”.
* CREATE MACHINE QTD - “CREATE MACHINE QTD\n”
  + Mensagem para criação das máquinas de guerra. MACHINE é uma string que indica qual a máquina que será construída. E QTD é um inteiro maior que 0, que indica a quantidade. São concatenados a string “CREATE”, separados por um espaço cada e finalizando a string com “\n”. Uma mensagem de confirmação é enviada, caso a solicitação de explorar seja aceita e uma mensagem de erro caso contrário.
  + Ex.: “CREATE FLYER 10\n”, etc.
* BET VALUE - “BET VALUE\n”
  + Mensagem para anúncio da quantia de aposta que um jogador fará para aquele conjunto de comandos. BET é uma string que é concatenada a VALUE, que por sua vez é um inteiro positivo que indica a quantia de Ether que será apostado. São separados por um espaço e concatenados a uma string “\n” para finalizar a mensagem. Aguarda uma mensagem de confirmação padrão ou de erro.
  + Ex.: “BET 0\n”, “BET 100\n”, etc.

4. Eventos de jogo

Existem trocas de informações entre o cliente e o servidor da instância do jogo que independem de ações do jogador para serem executadas. Estas se relacionam com o salvamento /carregamento do estado do jogo e envio de comandos por parte do servidor quando algum bloco é forjado. Com isto é possível manter os estados de jogo de todos os jogadores conectados bem próximo um do outro e manter a ordem de ações globalmente. As mensagens de evento de jogo seguem o padrão abaixo:

* SAVED GAMESTATE - “SAVED GAMESTATE\n”
  + Mensagem enviada pelo jogo para requerir o último estado do jogo (*game state*) salvo na Blockchain. Onde SAVED GAMESTATE é uma string de requerimento do último JSON de jogo salvo seguro da Blockchain. Uma mensagem de GAMESTATE (Seção x.y) é aguardada pelo cliente como resposta.
* GAMESTATE JSON - “GAMESTATE JSON\n”
  + Mensagem enviada pelo processo servidor contendo o último estado do jogo (*game state*) salvo. JSON é uma string que representa o estado do jogo, codificado no formato de json. Está é concatenada a string “GAMESTATE”, separados por um espaço e finalizando a mensagem com um “\n”. Não espera mensagem de confirmação em retorno.
* COMMAND PLAYER COMM - “COMMAND PLAYER COMM\n”
  + Mensagem enviada pelo servidor ao jogo sempre que um novo bloco é validado e adicionado a Blockchain. É enviada uma mensagem de command para cada comando presente em cada transação do novo bloco adicionado.

1. Comunicação servidor-servidor
2. Estabelecendo/Encerrando conexão

Todas as conexões com os servidores da rede são estabelecidas assim que a instância do servidor local é inicializada. Portanto ao iniciar o servidor local, este irá se conectar com os demais de sua lista de endereços (tentando manter 3 conexões) e manterá conexões persistentes TCP. A mensagem de estabelecimento e encerramento de conexão seguem os seguintes formatos:

* OPEN ID - “OPEN ID\n”
  + Onde “OPEN ID” é uma string finalizada com um fim de linha ‘\n’, que é a mensagem padrão para conexão com servidores. É esperado uma mensagem contendo a tabela de servidores de volta. Caso seja um novo jogador/servidor para entrar na rede o endereço IP é adicionado na tabela do servidor antes da tabela ser enviada. O cliente substituirá sua tabela de servidores local, caso a recebida seja maior. Em caso de erro, retorna erro.
* CLOSE - “CLOSE\n”
  + A string “CLOSE\n é enviada pelo servidor que deseja encerrar a conexão e não é esperada mensagem de confirmação.

1. Mensagem de confirmação/erro

Mensagem de confirmação enviada sempre depois de receber alguma requisição/entrega e de finalização de envio (OK/END), exceto no caso de encerramento CLOSE. E as mensagens de ERRO, quando algum problema ocorrer, sendo seguidas do código relacionado ao erro que ocorreu.

* OK - “OK\n”
  + Uma string “OK\n” que indica o recebimento correto da mensagem enviada anteriormente.
* END - “END\n”
  + Uma string “END\n” que indica que o último bloco da requisição já foi enviado, ou seja o fim do envio de blocos, em casos onde mais de um bloco é enviado. É formada por uma string “END” concatenada à um “\n”.
  + Ex.: “END\n”
* ERROR COD - “ERROR COD\n”
  + Onde COD representa um inteiro que está relacionado ao erro ocorrido. Sendo portanto concatenado na string “ERROR”, separados por um espaço e finaliza a mensagem com “\n”.
  + Ex.: “ERROR 0\n”, “ERROR 127\n”, “ERROR 7\n”, etc.

1. Requisição de blocos ou toda a Blockchain

As mensagens de requisição de blocos são enviadas para servidores conectados, sempre que um servidor que estava desligado se conecta a rede, um servidor novo entra na Blockchain, ou uma cadeia de blocos foi ultrapassada. Os blocos são enviados um a um e para cada recebimento de bloco uma mensagem de confirmação (OK) é enviada pelo servidor que fez a requisição. Ao final, o servidor que está enviando os blocos, envia uma mensagem de finalização de envio (END) para sinalizar que todos os blocos requisitados já foram enviados e encerrar aquela requisição.

* BLOCK ALL - “BLOCK ALL\n”
  + Uma string única “BLOCK ALL\n” onde o “ALL”, separado por um espaço, indica que todos os blocos são requeridos.
* BLOCK SINCE I - “BLOCK SINCE I\n”
  + A string “BLOCK SINCE” é concatenada, juntamente com um espaço em branco, à um inteiro I (“i”) seguido de fim de linha. Este inteiro I representa o index do último bloco que o servidor que fez a requisição possui. Todos os blocos à partir deste index, até o último já validado pela rede serão enviados.
  + Ex: “BLOCK SINCE 30\n”, “BLOCK SINCE 0\n”, “BLOCK SINCE 123\n”, etc.
* BLOCK BEFORE I - “BLOCK BEFORE I\N”
  + A string “BLOCK BEFORE” é concatenada, juntamente com um espaço em branco, à um inteiro I (“i”) seguido de fim de linha. Este inteiro I representa o index do maior bloco atual que o servidor possui, e é usada quando algum servidor tem sua cadeia mais longa ultrapassada e necessita recuperar blocos da nova cadeia principal. Um bloco de index I - 1 é enviado ao servidor que fez a requisição.
  + Ex: “BLOCK BEFORE 30\n”, “BLOCK BEFORE 200\n”, “BLOCK BEFORE 123\n”, etc.
* BLOCK I - “BLOCK I\n”
  + Mensagem enviada quando um requerimento de blocos é feita, pelo servidor que está atendendo a requisição. Onde BLOCK é um vetor de bytes que representa a codificação do bloco. Finaliza a mensagem com um “\n”.

1. Disseminação de comandos e blocos

Mensagens de disseminação de comandos e blocos são aquelas que serão enviadas por meio de mensagens broadcast para toda a rede quando ocorrerem. O método utilizado para o envio é a inundação, garantindo que todos os nós que possuem conexão vigente receberão a mensagem, apesar de aumentar significativamente o tráfego da rede.

O tratamento de mensagens repetidas é realizado de duas formas, na mensagem de vitória (WIN BLOCK), não é feita nenhuma tratativa, uma vez que cada bloco possui um número de identificação único e sequencial, logo mensagens com blocos de index repetidos serão automaticamente descartadas. Nas transações foi colocado na estrutura da mensagem um campo de número de sequência (SeqNumber) e do Player que enviou a mensagem, logo mensagens com o SeqNumber menor que o último recebido, para aquele player, serão consideradas repetidas e descartadas.

* WIN BLOCK\n - “WIN BLOCK\n”
  + BLOCK é um vetor de bytes que representa a codificação do bloco. Finaliza a mensagem com um “\n”.
* COMMAND PLAYER MSG\n - “COMMAND PLAYER MSG\n”
  + PLAYER é uma string que representa o *login* do player que realizou o comando. MSG é uma string que indica o comando realizado pelo player, que segue alguma das especificações descritas na seção I.2 deste documento. Finaliza a mensagem com um “\n”. Espera uma mensagem de confirmação padrão.
* TRANSACTION JSON\n - “TRANSACTION JSON\n”
  + Mensagem de espalhamento de transações. Esta é enviada sempre que um servidor recebe uma nova transação de um servidor conectado, ou cria uma com os comandos recebidos do jogo. JSON é uma string que representa a codificação de uma transação em formato JSON, seguindo o padrão de transações abordados em (z.a). Esta codificação é concatenada ao final da string "TRANSACTION", separados por um espaço e concatenado a um "\n" para finalizar a mensagem. É aguardado um ACK de confirmação padrão ou erro como retorno.