

**UNIVERSIDADE FEDERAL DE PELOTAS – UFPEL**

**Engenharia de Computação**



**Redes de Computadores**

**Trabalho Prático**

**Dara dos Santos Lima - 16103611**

**Géron Wilham da Cruz Mattoso - 16100886**

## Jogo da Velha

O presente trabalho consiste no desenvolvimento de um jogo da velha, na linguagem Python, utilizando os conceitos abordados durante a disciplina sobre redes, principalmente aqueles que envolvem a camada de aplicação.

O jogo é composto por dois jogadores, em que um é o cliente e o outro o servidor. O servidor aguarda até que o jogador correspondente se conecte para dar início ao jogo. Portanto, foi utilizada a arquitetura cliente-servidor. O jogo roda em ambas as máquinas que trocam mensagens entre si a cada jogada, atualizando o tabuleiro utilizado durante a partida.

## Características do Protocolo Desenvolvido

O protocolo utilizado foi o TCP, uma vez que para este tipo de aplicação a preocupação maior está voltada em manter a integridade dos dados, no caso do tabuleiro do jogo principalmente, e não relacionada a velocidade de envio e recepção destes.

A aplicação é sem estado, uma vez que não armazena informações de conexões anteriores, isto é, de partidas passadas. Além disso, o protocolo desenvolvido é do tipo push, onde não temos uma solicitação das informações, elas são apenas enviadas para que cada máquina faça suas atualizações pertinentes ao funcionamento do jogo. A conexão utilizada é persistente, uma vez que a partida ocorre dentro de uma única conexão TCP, a conexão só é encerrada ao final do jogo, quando alguém vencer ou resultar em empate. O protocolo desenvolvido possui controle na banda, já que em uma mesma conexão TCP são passadas informações de controle e de dados do jogo.

## Funcionamento Geral e Troca de Mensagens

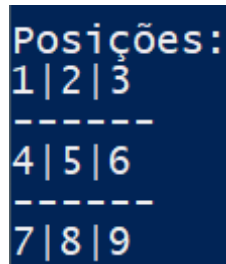
Inicialmente é necessário que o servidor seja executado, este esperará pela conexão (TCP) de um cliente que será o segundo jogador. Para que a partida comece é preciso que o cliente entre com a mensagem “start”, que será enviada para o servidor indicando que o jogo pode iniciar. Depois disso, o servidor identifica o IP do cliente e enviará para ele, a fim de manter a identificação do segundo jogador e o tabuleiro inicial do jogo vazio, representado pela string: `board = 123456789`.

O funcionamento do jogo é baseado em trocas de mensagens entre os jogadores (cliente e servidor). Cada mensagem, no formato de uma string, armazena uma codificação que corresponde ao tabuleiro do jogo, como o exemplo a seguir.

A string: `board = 12X456O89` corresponde ao seguinte tabuleiro:

```
Tabuleiro:
| |X
-----
| |
-----
O| |
```

A decodificação da string é realizada com base nas posições possíveis do tabuleiro, que são informadas aos jogadores, como mostra a figura a seguir.



Posições:

1	2	3
---	---	---
4	5	6
---	---	---
7	8	9

Desta forma, os números da string que representa o tabuleiro indicam as posições livres. A cada jogada realizada é adicionado o símbolo do jogador(X ou O) na posição escolhida, que é inclusa na string que é enviada para o próximo jogador. Ambos os jogadores, cliente e servidor, trocam este mesmo tipo de mensagem diversas vezes, a cada movimentação realizada.

Além disso, existem outros tipos de mensagens que são trocadas no momento em que se tem um vencedor ou ocorre um empate. O jogador que efetua a jogada que vence a partida envia uma mensagem para seu correspondente contendo a string “win”, que indica que este venceu o jogo. Quando uma mensagem “win” é recebida, pode-se saber que o oponente venceu o jogo, logo o jogo é encerrado tanto no cliente como no servidor, indicando a ambos o jogador vencedor identificado por seu símbolo (X ou O) e número de IP.

A mensagem que contém a string “emp” é utilizada para representar que ocorreu um empate na partida. O primeiro jogador a constatar o empate envia a mensagem informando ao oponente que o jogo encerrou e que seu resultado foi um empate.

#### **Listagem/Formato dos tipos de mensagens possíveis do cliente para o servidor:**

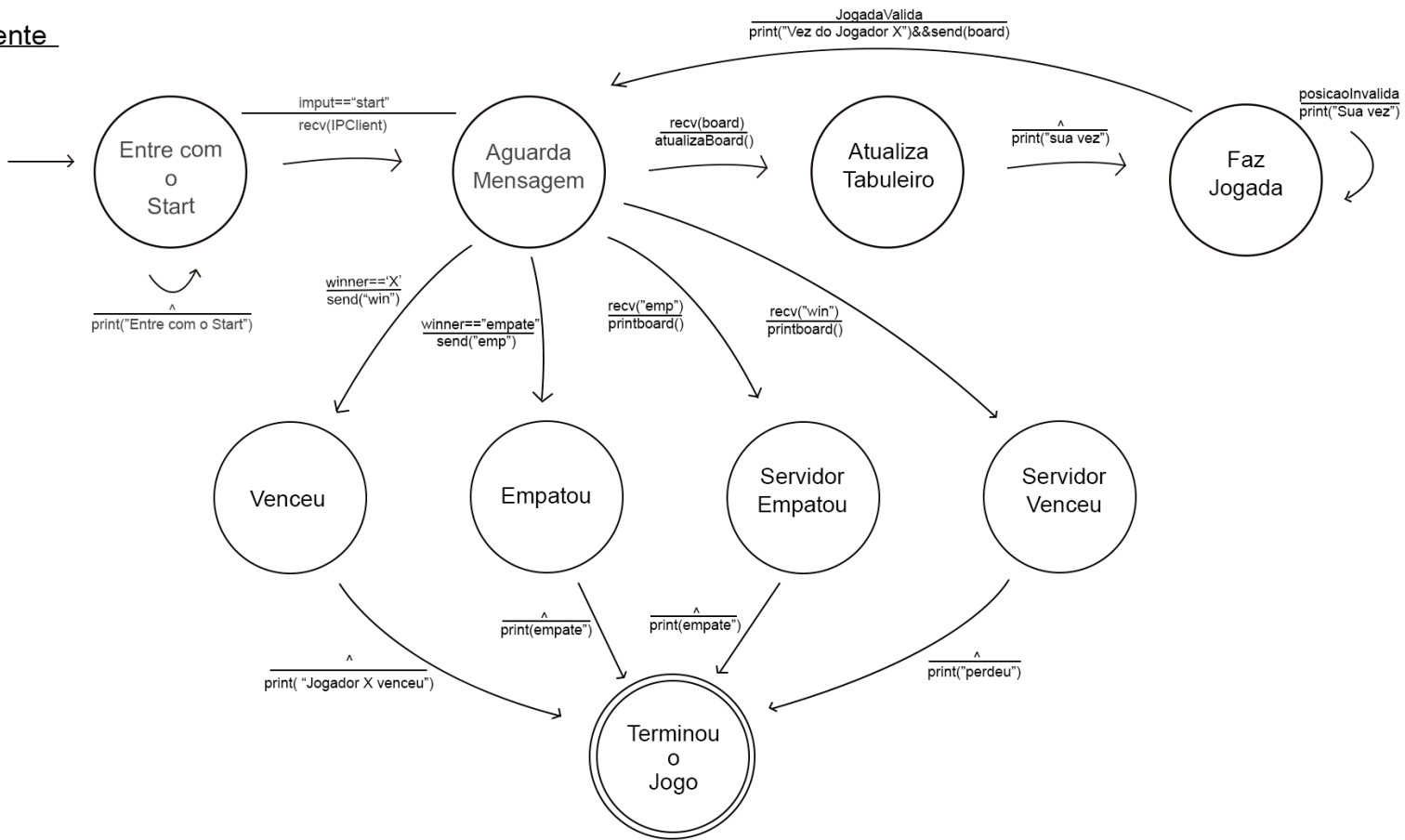
- Início do jogo: message = ‘start’.
- Tabuleiro do jogo, do tipo: board = ‘12X4OO78X’.
- Venceu a partida: ‘win’.
- Empatou: ‘emp’.

#### **Listagem/Formato dos tipos de mensagens possíveis do servidor para o cliente:**

- IP do cliente, para identificação: clientAddress[0].
- Tabuleiro do jogo, do tipo: board = ‘12X4OO78X’.
- Venceu a partida: ‘win’.
- Empatou: ‘emp’.

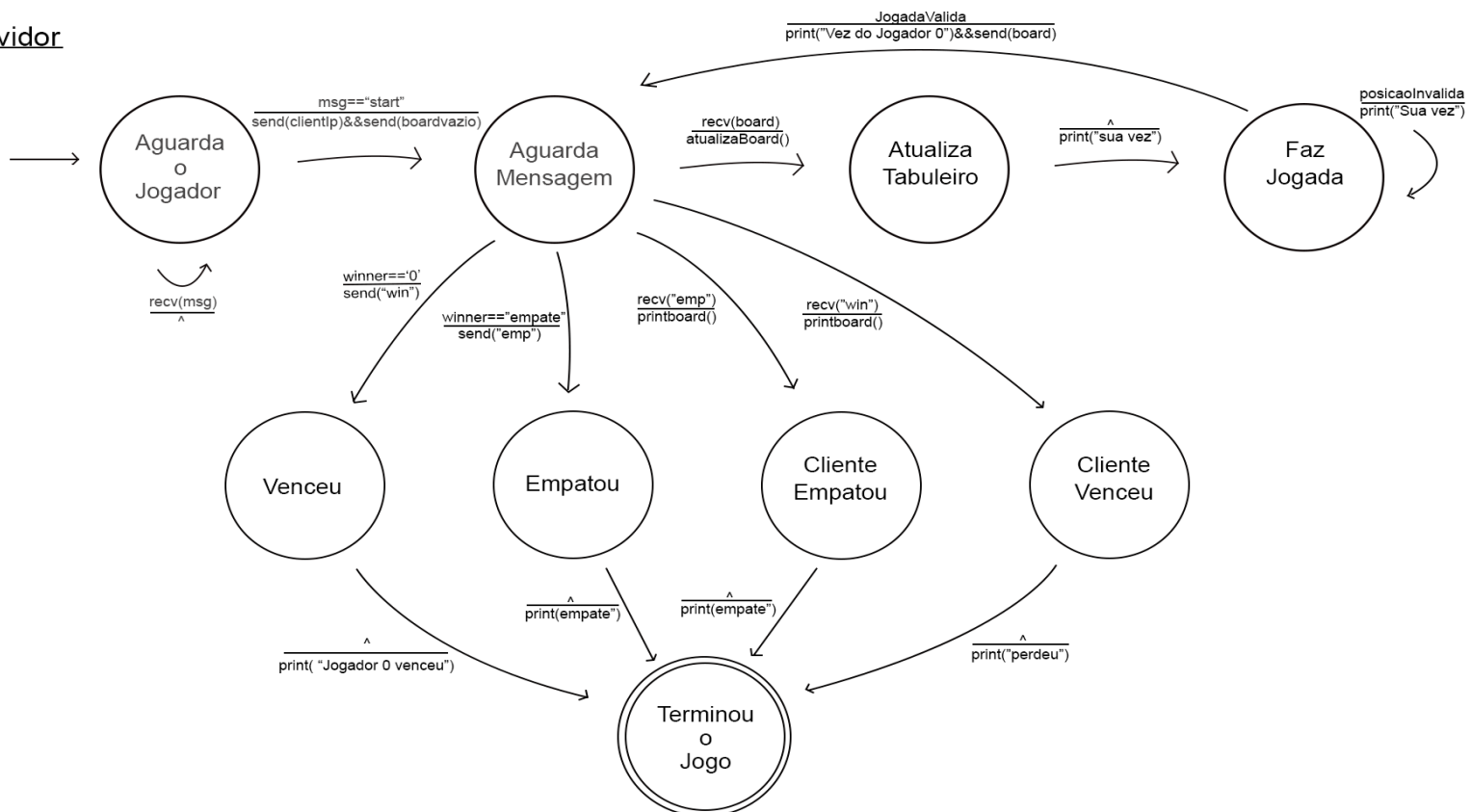
## Máquina de estados do cliente para o protocolo desenvolvido

### Cliente



## Máquina de estados do servidor para o protocolo desenvolvido

### Servidor



É importante considerar que alguns detalhes foram abstraídos em ambas as máquinas de estados. Optou-se por demonstrar o essencial, e lógica geral que possibilitam o funcionamento do jogo desenvolvido. Além disso, algumas funções não existem com os nomes apresentados, como o caso de “PosiçãoInválida”, este nome foi dado visando facilitar o entendimento e a percepção geral da funcionalidade que é a mesma desenvolvida no código porém com o uso de outras funções e ações.

### **Dificuldades Encontradas**

A ideia inicial, pensada para o desenvolvimento do trabalho, era utilizar um servidor e dois clientes. O servidor seria responsável apenas por efetuar as atualizações de tabuleiro e controle das jogadas, enquanto os clientes seriam os jogadores. Após algumas tentativas e muitos erros encontrados acabamos desistindo, já que para isso é necessário trabalhar com threads e nossa dupla não possui muita familiaridade com relação ao seu uso.

Além disso, seria interessante desenvolver uma interface para o jogo, o que acabamos não conseguindo concluir a tempo, mas ainda pretendemos terminar.

### **Conclusão:**

Diante do desenvolvimento do presente trabalho foi possível compreender na prática os conceitos abordados em teoria durante as aulas, o que proporcionou melhor fixação dos conteúdos e entendimento. O jogo desenvolvido apesar de possuir uma estrutura simples apresentou funcionamento como o esperado e propiciou reflexões e estudos importantes.

Além disso, a realização da prática promoveu a curiosidade sobre o desenvolvimento de aplicações que utilizam estrutura de rede, seu funcionamento, e com isso vontade de desenvolver novas aplicações até mesmo mais complexas quando tivermos mais tempo disponível.