UVG UNIVERSIDAD DEL VALLE DE GUATEMALA

SHINE Protocol

PROTOCOLO SHINE-SERVER
"JUEGO DE CARTAS - OREJITAS"

Guatemala, 15 de septiembre de 2021

Elaborado por:

María Mercedes Retolaza

Cesar Rodas

André Rodríguez

INTRODUCCIÓN

La interconexión de sistemas o redes de computadoras son la base de las comunicaciones hoy en día y están diseñadas bajo múltiples protocolos de comunicación. Existen muchos protocolos al establecer una conexión a internet y según el tipo que se necesite establecer, dichos protocolos van a variar. Además, la comunicación a internet no es el único tipo de comunicación cuando hablamos de transmisión de datos e intercambio de mensajes en redes. En todos los casos, los protocolos de red definen las características de la conexión.

Un protocolo es un conjunto de reglas: los protocolos de red son estándares y políticas formales, conformados por restricciones, procedimientos y formatos que definen el intercambio de paquetes de información para lograr la comunicación entre dos servidores o más dispositivos a través de una red.

Los protocolos de red incluyen mecanismos para que los dispositivos se identifiquen y establezcan conexiones entre sí, así como reglas de formato que especifican cómo se forman los paquetes y los datos en los mensajes enviados y recibidos. Algunos protocolos admiten el reconocimiento de mensajes y la compresión de datos diseñados para una comunicación de red confiable de alto rendimiento. En el presente proyecto se establecerá el protocolo de conexión para la transferencia de información que requiere un juego de cartas básico. A continuación, se establece el alcance, requerimientos y definición del protocolo para poder llevar el control y seguimiento de un juego de cartas.

OBJETIVO GENERAL

Establecer un protocolo de conexión y comunicación **basado en TCP** para establecer el estándar de interacción que debe de tener un juego de cartas que permita la interacción de un mínimo de **3 jugadores.**

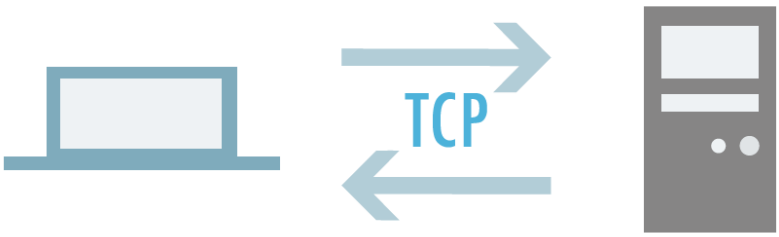
Objetivos Específicos

- Establecer las reglas e instrucciones que deberá de seguir el protocolo de conexión
- Crear un juego interactivo de cartas que permita un mínimo de 3 jugadores
- Crear y configurar un servidor para la ejecución e interacción del juego

PROTOCOLO BASADO EN TCP

¿Qué es TCP?

Protocolo de control de transmisión (TCP) es uno de los protocolos fundamentales en Internet. Fue creado entre los años 1973 y 1974 por Vint Cerf y Robert Kahn. Es un protocolo de internet encargado de informar del destino de los datos permitiendo la creación de conexiones seguras. Aunque fue desarrollado entre 1973 y 1974, continúa siendo a día de hoy uno de los protocolos fundamentales en internet. TCP sirve, además, como soporte a muchas de las aplicaciones y protocolos que han surgido después.



El protocolo TCP es una forma segura de intercambio de datos al requerir de la autorización entre cliente y servidor, o emisor y receptor, antes de producirse la transferencia. Una vez ambas partes hayan autorizado la transmisión, podrá iniciarse el envío y recepción de datos. El protocolo TCP, al igual que otros como el protocolo SSH, nació para sustituir protocolos anteriores; debido a su antigüedad, dichos protocolos resultaban inseguros para la conexión o el intercambio de datos en internet.

¿Cómo funciona el protocolo TCP?

El funcionamiento consta de tres fases. La primera, se establece la conexión con la autorización de ambas partes. Entonces, se produce un procedimiento denominado “negociación en tres pasos”. Después, se inicia la transferencia de información y aquí se establecen cada uno de los parámetros

para un intercambio ordenado, correcto y, sobre todo, seguro. Por último, mediante una negociación en cuatro pasos se finaliza la conexión entre el cliente y servidor.

En el protocolo TCP los datos se entregan en el mismo orden en el que se enviaron. Para ello, divide la información en diferentes paquetes que se envían por la ruta más rápida hacia su destino. Así, con una separación en capas, se identifica la procedencia del tráfico es más fácil y evitar la saturación de la red. Además, sirve de capa intermedia entre una aplicación y el protocolo IP, supliendo las carencias de seguridad del protocolo de red (consulta aquí cuál es tu dirección IP).

TCP sirve también como mecanismo que permite diferenciar las aplicaciones, ya sean emisoras o receptoras, dentro de una misma máquina. Para ello, recurre al concepto de puerto. A pesar de tener ya más de cuatro décadas, sigue empleándose en todas las comunicaciones que se producen en la red de redes: internet. Y, sin su desarrollo, el internet que conocemos actualmente sería muy diferente.

ALCANCE

El objetivo del proyecto es **establecer un protocolo de conexión** basado en TCP que sea el encargado de **establecer la comunicación** entre **cliente y servidor** para la elaboración de un juego de cartas básico llamado **Orejitas.** Se deberá de crear un sistema que permita llevar el control del juego. Deberá de permitir ingresar a una plataforma web en donde el cliente(s) podrá interactuar con el juego según las reglas establecidas por el autor.

El juego podrá iniciarse **si y solo sí** se tiene un mínimo de **tres jugadores** y deberá de tener un **máximo permitido de 7 jugadores por juego.** El juego **no deberá de almacenar información personal del cliente por seguridad** por lo que deberá de ingresar al sitio web y crear una sala o ingresar un **ID único de sala** para poder **unirse al juego** e iniciar una partida.

Los jugadores podrán recibir un máximo de **5 cartas** y, cuando se tiene la **totalidad de jugadores disponibles,** la mesa se quedará con un total de **17 cartas** para que los jugadores puedan jugar e interactuar entre cada uno.

Los jugadores podrán iniciar el juego **siempre y cuando** se cumpla con el **mínimo de jugadores de la sala establecido** para el juego, el primer tiro se elige de forma aleatoria y las cartas deberán de ser repartidas de forma aleatoria. El mazo que se encuentra en la mesa se deberá de mezclar de forma aleatoria antes de iniciar la partida y repartir las cartas. (No existe un patrón).

El juego deberá de establecer la **primera carta** como la carta **patrón** para poder lanzar las siguientes cartas, si un jugador **tiene la capacidad de cambiar de juego** deberá de contener entre su mano una carta de la misma **figura y mayor** a la que se encuentra en la mesa para poder elegir **el nuevo juego**, en este momento el jugador podrá lanzar dos cartas sobre la mesa. El servidor deberá de tomar esa nueva carta y actualizar el juego completo.

Se establece un ganador cuando se quede sin cartas, los demás jugadores serán automáticamente los perdedores del juego. Es importante que, cuando la mesa se quede sin cartas para poder transferir información o cambiar el juego, este deberá de terminar de forma **automática** y se hará un resumen del juego.

REQUERIMIENTOS FUNCIONALES

Acciones del juego

- Crear una sala
    - Seleccionar un máximo de jugadores
    - Ingresar mi nickname
        - Administrador de la sala
    - Generar un ID para que los demás usuarios puedan unirse a esa sala
- Unirse a una sala
    - Ingresar ID
        - Verificar que la sala sea existente
        - Verificar que la sala no contenga el número máximo de jugadores
    - Ingresar nickname
- Iniciar un juego
    - Seleccionar a primer jugador
    - Lanzar una carta (primer jugador)
        - Establecer figura de juego

- Notificación → Es tu turno
- Cambiar de juego
- ¿Quién gana?
    - La mesa se queda sin cartas
        - Gana el jugador que tiene la menor cantidad de cartas
    - Un jugador coloca su última carta
        - Gana el jugador que en su mano tiene 0 cartas
    - Se cierra el juego por conexión a internet
        - Gana el jugador que tiene la menor cantidad de cartas
- Saludar a compañeros de la sala → Mensajes de grupo
    - Enviar mensaje
    - Recibir mensaje
    - Anunciar ingreso de nuevo compañero a una sala
- Salir del juego

Definición del protocolo “Shine”

- ¿Qué es un Sparks?
    - Un Sparks es el encargado de intercambiar información entre el servidor “Shine” y un cliente. Existen varios tipos de Sparks que nos ayudan a estructurar y definir la información que deberemos de transmitir a través de la interacción que se tenga con el cliente y el servidor.
    - **Asincronismo:** El protocolo Shine funciona de forma asíncrona, las conexiones se establecen durante mucho tiempo y en cualquier momento el servidor y /o el cliente pueden enviar y recibir **Sparks** a través de este canal.
- ¿Cuáles tipos de Sparks existen?
    - Los tipos de Sparks son los ID de eventos que estaremos escuchando para poder saber que acción emitir. Debido a eso, los eventos tendrán un identificador que nos permita asociar las acciones o interacciones de las diferentes dinámicas y funcionalidades que contiene nuestro juego.
        - Eventos
            - createRoom
                - Es el evento que nos indica que una nueva sala existe en nuestro servidor, se genera un evento que nos permite almacenar ese ID entre los valores existentes y nos

permite compartir ese ID para que otros usuarios se conecten

- joinRoom
  - Nos permite asignar a un usuario a una sala
- chatRoom
  - Nos permite la interacción de mensajes entre los usuarios por medio de una sala
- startRoom
  - Corresponde al evento que nos indica que el juego está por comenzar
- takeCard
  - Corresponde al evento que se encarga de conocer los movimientos de mano que hacen los usuarios, nos ayuda a conocer en que momento debemos de actualizar nuestros contadores y mesas
- makeMove
  - Corresponde al evento que se encarga de realizar y controlar los movimientos del cliente
- disconnect
  - Corresponde al evento que se encarga de desconectar al cliente del room/servidor

- Documentación del protocolo
    - Crear una sala
        - Envío del servidor

```
// respuesta para crear una sala
// el server lo emite a traves del evento createRoom
{
  sparkId: 1,
  type: 'response',
  success: true,
  data: {
    createdId: sala,
    roomdeck: numero,
    tablecard: null
  },
};
```

        - Respuesta del cliente

```
// crea una sala
{
  type: 'createRoom',
  data: {
    name: texto,
    description: texto,
    maxPlayers: numero,
    userNickname: texto
  },
}
```

- Unirse a una sala
  - Respuesta del servidor

```
// respuesta para ingresar en una sala
// el server lo emite a traves del evento joinRoom
{
  sparkId: 2,
  type: 'response',
  success: false,
  data: {
    message: mensaje,
  },
};


{
  sparkId: 2,
  type: 'response',
  success: true,
  data: {
    roomid: sala,
    name: texto,
    description: texto,
    message: texto,
    players: array con info de jugadores,
  },
};
```

  - Respuesta del cliente

```
// ingresa a una sala
{
  type: 'joinRoom',
  data: {
    roomId: texto,
    nickname: texto
  },
};
```

- Iniciar un juego
  - Respuesta del servidor

```
// respuesta para iniciar el juego en una sala
// el server lo emite a traves del evento startRoom
{
  sparkId: 4,
  type: 'response',
  success: false,
  data: {
    message: text,
  },
};
```

- Si la respuesta es positiva

```
// si la respuesta es positiva se emite de forma individual a cada
// jugador en la sala por el evento startRoom
// cada jugador podrá ver unicamente sus cartas,
// en el listado de los otros jugadores las cartas de los demás las recibe null
{
  sparkId: 4,
  type: 'response',
  success: true,
  data: {
    start: boolean,
    players: [
      {
        name: text,
        socket: text,
        room: text,
        cards: [
          {
            num: numero,
            fig: numero,
            img: texto
          },
        ],
        countCards: numero,
        turn: boolean
      }
    ],
    roomdeck: numero,
    tablecard: {
      num: numero,
      fig: numero,
      img: texto
    },
  },
};
```

- Respuesta del cliente

```
// inicia una partida en una sala
{
  type: 'startRoom',
  data: {
    roomId: texto,
    nickname: texto
  },
};
```

- Movimientos del juego
  - Tomar una carta del mazo
    - Respuesta del cliente

```
// toma una carta del maso y se la agrega al jugador
{
  type: 'takeCard',
  data: {
    roomId: texto,
    nickname: texto
  },
};
```

    - Respuesta del servidor

```
// respuesta para tomar una carta del mazo, actualizando a todos los jugadores
// el server lo emite a traves del evento takeCard
{
  sparkId: 5,
  type: 'response',
  success: false,
  data: {
    message: texto,
  },
};
```

- Movimientos
  - Respuesta del cliente

```
// hacer un movimiento y/o cambiar de color
{
  type: 'makeMove',
  data: {
    roomId: texto,
    nickname: texto,
    card: {
      num: numero,
      fig: numero,
      img: texto
    },
    change: boolean,
    change_card: {
      num: numero,
      fig: numero,
      img: texto
    }
  }
}
```

  - Respuesta del servidor: En los movimientos nos encargamos de actualizar los eventos de los jugadores que se encuentran con un juego iniciado. Para ello manejamos las siguientes respuestas al cliente para que realice las actualizaciones del juego y muestra el control que lleva el servidor de los movimientos.

```
// la respuesta se emite de forma directa  a cada jugador
// en la sala, impidiendo que puedan ver las cartas de los demás
{
  sparkId: 5,
  type: 'response',
  success: true,
  data: {
    start: boolean,
    players: [
      {
        name: text,
        socket: text,
        room: text,
        cards: [
          {
            num: numero,
            fig: numero,
            img: texto
          },
        ],
        countCards: numero,
        turn: boolean
      }
    ],
    roomdeck: numero,
    tablecard: {
      num: numero,
      fig: numero,
      img: texto
    },
  },
};
```

```
// respuesta realizar un movimiento
// el server lo emite a traves del evento startRoom
{
  sparkId: 6,
  type: 'response',
  success: false,
  data: {
    message: mensaje,
  },
};
```

```
// la respuesta se emite de forma directa
// a cada jugadore en la sala, impidiendo que puedan ver las cartas de los demás
{
  sparkId: 6,
  type: 'response',
  success: true,
  data: {
    start: boolean,
    players: [
      {
        name: text,
        socket: text,
        room: text,
        cards: [
          {
            num: numero,
            fig: numero,
            img: texto
          },
        ],
        countCards: numero,
        turn: boolean
      }
    ],
    roomdeck: numero,
    tablecard: {
      num: numero,
      fig: numero,
      img: texto
    },
  },
};
```

- Mensajes
  - Recepción/Envío de mensaje
    - Respuesta del servidor

```
// respuesta para chatear en una sala
// el server lo emite a traves del evento chatRoom
{
  sparkId: 3,
  type: 'response',
  success: true,
  data: res.data,
};
```

    - Respuesta del cliente

```
// enviar mensajes al chat de la sala
{
  type: 'chatRoom',
  data: {
    roomId: texto,
    nickname: texto,
    message: texto
  },
};
```

Características del servidor

- Permitir la creación de varias salas
- Permitir juegos concurrentes
- Llevar el control del juego
    - Manejo de turnos
    - Cambio de juego
    - Envío de mensajes
    - Asignaciones a salas

REQUERIMIENTOS NO FUNCIONALES

Los requerimientos no funcionales representan características generales y restricciones de la aplicación o sistema que se esté desarrollando. Por lo que en nuestro juego de cartas: **Orejitas** cumpliremos los siguientes requisitos:

Eficiencia

- Toda funcionalidad del sistema y transacción de negocio debe de responder al usuario en un tiempo eficaz.
- El servidor debe de indicar los movimientos de los jugadores en tiempo real

Seguridad lógica y de datos

- El nuevo sistema debe desarrollarse aplicando patrones y recomendaciones de programación que incrementan la seguridad de los datos
- La información transmitida a través de Sockets deberá de estar cifrada para resguardar la integridad de los jugadores
- La transferencia de mensajes deberá de ser cifrada

Usabilidad

- El tiempo de aprendizaje del sistema por usuario deberá de ser menor a 4 horas
- La tasa de errores cometidos por usuario deberá de ser menor al 5% de las transacciones totales ejecutadas en el sistema
- El diseño debe de ser adaptable para las resoluciones más utilizadas de aplicaciones móviles de hoy en día

DIFICULTADES DEL PROYECTO

- Definición del protocolo

  Al momento de definir los Sparks que se encargarán de interactuar con el cliente y el servidor nos topamos en que debíamos conocer a profundidad los requerimientos para poder establecer los JSON que iban a utilizarse en el intercambio de información entre los sockets

  - ¿Cómo lo establecemos?
    - Analizamos el alcance y las reglas de juego para poder evaluar la transferencia de información entre cliente y servidor
    - Analizamos la mecánica de interacción de información entre sockets
    - Pensamos de forma asíncrona, pues no tendremos la capacidad de conocer si se activo o no un proceso, tenemos que escuchar a nuestro servidor para poder determinar la respuesta

        - Realizamos un orden, establecimos la configuración que deben de tener nuestras Sparks asignando un tipo para poder **lanzar una acción en especifica** cuando el servidor detecte que se disparó el evento
    - ¿Cómo realizamos una estructura correcta?
        - Realizamos una estructura en papel
        - Iniciamos con la configuración del servidor
            - Handler
            - Brodcast
            - Funciones básicas y repetitivas
                - GetUsername
                - GetRoomID
- Configuración del servidor
    - Se realizó una configuración del servidor en dos puertos, ya que el 2222 que está destinado a TCP no pudo configurarse debido a las propiedades del servidor de pruebas. Por lo que se configuraron los siguientes puertos
        - WebSockets: 5000
        - TCP: 5222
- Conexión del cliente con el servidor
    - Al momento de establecer la conexión con el cliente y el servidor, tuvimos que aplicar correcciones a la mecánica / estructura actual del servidor. Pues, realizamos una integración de la librería de Socket.io para poder hacer uso de los WebSockets y así poder interactuar con nuestro cliente en Angular

FUETES CONSULTADAS

TCP (Transmission Control Protocol): retrato del protocolo de transporte. (2021, 17 septiembre). IONOS Digitalguide. https://www.ionos.es/digitalguide/servidores/know-how/que-es-tcp-transport-control-protocol/

Protocolos TCP/IP. (2021). IBN - Protocolos TCP/IP. https://www.ibm.com/docs/es/aix/7.1?topic=protocol-tcpip-protocols