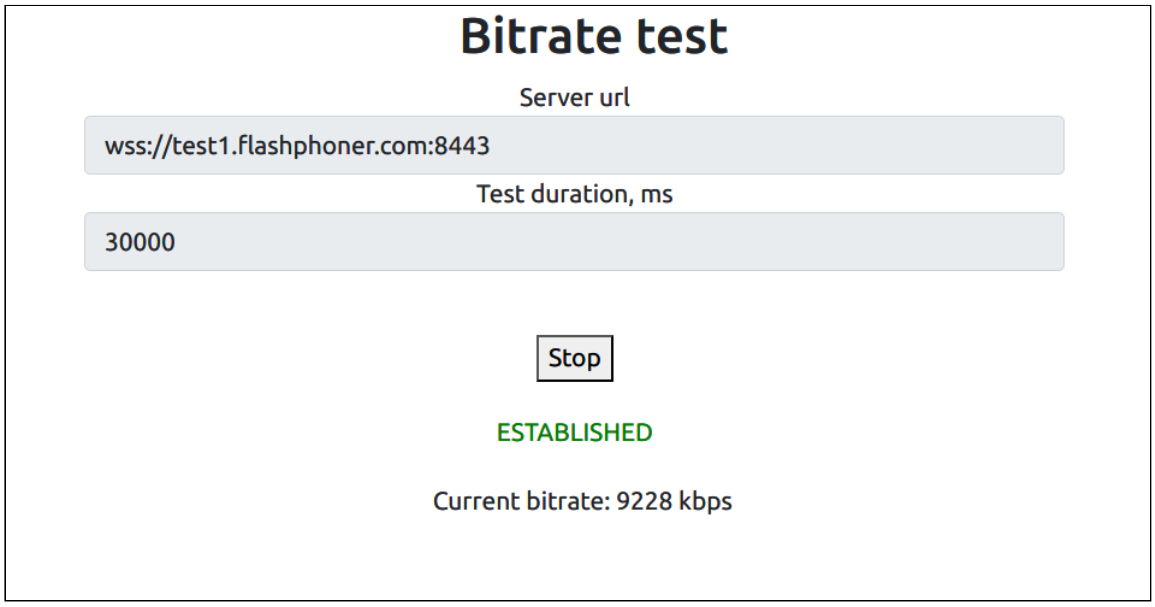
SFU Bitrate Test

Пример демонстрирует тестирование пропускной способности канала публикации

На скриншоте ниже

- `Server url` - Websocket URL WCS сервера
- `Test duration` - максимальная длительность теста в мс
- `Current bitrate` - текущий измеренный битрейт передачи данных в кбит/с



Исходный код примера

Исходный код разбит на следующие модули:

- `bitrate_test.html` - HTML страница
- `bitrate_test.css` - стили HTML страницы
- `bitrate_test.js` - основная логика приложения

Анализ исходного кода

Для работы с исходным кодом примера возьмем версию файла `bitrate_test.js`, доступную здесь.

1. Локальные переменные

Объявление локальных переменных для работы с константами, SFU SDK, для хранения текущего состояния и работы с конфигурацией тестовой комнаты

code

```
const constants = SFU.constants;
const sfu = SFU;
let bitrateTestState;

const BITRATE_TEST = "bitrateTest";
const TEST_DURATION = 30000;
```

2. Объект для хранения текущего состояния тестирования

Хранит данные Websocket сессии, WebRTC соединения, SFU комнаты и объекта для запуска теста

code

```
const CurrentState = function (prefix) {
    let state = {
        prefix: prefix,
        pc: null,
        session: null,
        room: null,
        bitrateController: null,
        set: function (pc, session, room) {
            state.pc = pc;
            state.session = session;
            state.room = room;
        },
        clear: function () {
            state.room = null;
            state.session = null;
            state.pc = null;
            state.bitrateController = null;
        },
        durationId: function () {
            return state.prefix + "Duration";
        },
        buttonId: function () {
            return state.prefix + "Btn";
        },
        statusId: function () {
            return state.prefix + "Status";
        },
        errInfoId: function () {
            return state.prefix + "ErrorInfo";
        },
        currentStateId: function () {
            return state.prefix + "CurrentState";
```

```
        },
        getBitrateController: function () {
            return state.bitrateController;
        },
        setBitrateController: function (controller) {
            state.bitrateController = controller;
        },
        isConnected: function () {
            return (state.session && state.session.state() ===
constants.SFU_STATE.CONNECTED);
        }
    };
    return state;
}
```

3. Инициализация

code

Функция `init()` вызывается после того, как страница загрузится:

- инициализирует объекты состояния
- инициализирует поля ввода

```
const init = function () {
    bitrateTestState = CurrentState(BITRATE_TEST);
    $("#" + bitrateTestState.buttonId()).prop('disabled', true);
    $("#url").prop('disabled', true);
    onDisconnected(bitrateTestState);
    $("#url").val(setURL());
    $("#" + bitrateTestState.durationId()).val(TEST_DURATION);
}
```

4. Соединение с сервером

`RTCPeerConnection()`, `SFU.createRoom()` code

Функция `connect()` вызывается по нажатию кнопки `Start`:

- создает объект `PeerConnection`
- очищает отображение статуса предыдущей сессии
- настраивает конфигурацию комнаты и создает Websocket сессию
- подписывается на события Websocket сессии

```
const connect = async function (state) {
    //create peer connection
    const pc = new RTCPeerConnection();
    //get config object for room creation
    const roomConfig = getRoomConfig(defaultConfig);
    roomConfig.url = $("#url").val();
```

```
    roomConfig.roomName = "ROOM1-" + createUUID(4);
    roomConfig.nickname = "User1" + createUUID(4);
    // clean status display items
    setStatus(state.statusId(), " ");
    setStatus(state.errInfoId(), " ");
    // clean bitrate display item
    $("#" + state.currentStateId()).val("");
    // connect to server and create a room if not
    try {
        const session = await sfu.createRoom(roomConfig);
        // Set up session ending events
        session.on(constants.SFU_EVENT.DISCONNECTED, function () {
            onStopClick(state);
            onDisconnected(state);
            setStatus(state.statusId(), "DISCONNECTED", "green");
        }).on(constants.SFU_EVENT.FAILED, function (e) {
            onStopClick(state);
            onDisconnected(state);
            setStatus(state.statusId(), "FAILED", "red");
            if (e.status && e.statusText) {
                setStatus(state.errInfoId(), e.status + " " + e.statusText,
"red");
            } else if (e.type && e.info) {
                setStatus(state.errInfoId(), e.type + ": " + e.info, "red");
            }
        });
        // Connected successfully
        onConnected(state, pc, session);
        setStatus(state.statusId(), "ESTABLISHED", "green");
    } catch (e) {
        onDisconnected(state);
        setStatus(state.statusId(), "FAILED", "red");
        setStatus(state.errInfoId(), e, "red");
    }
}
```

5. Запуск тестирования канала при установке соединения

code

Функция `onConnected()`:

- настраивает действия по нажатию кнопки `Stop`
- подписывается на события об ошибках и завершении комнаты
- вызывает функцию тестирования

```
const onConnected = function (state, pc, session) {
    state.set(pc, session, session.room());
    $("#" + state.buttonId()).text("Stop").off('click').click(function () {
        onStopClick(state);
    }).prop('disabled', false);

    $('#url').prop('disabled', true);
    $("#" + bitrateTestState.durationId()).prop('disabled', true);
```

```
    // Add errors displaying
    state.room.on(constants.SFU_ROOM_EVENT.FAILED, function (e) {
        setStatus(state.errInfoId(), e, "red");
        onStopClick(state);
    }).on(constants.SFU_ROOM_EVENT.OPERATION_FAILED, function (e) {
        onOperationFailed(state, e);
    }).on(constants.SFU_ROOM_EVENT.ENDED, function () {
        setStatus(state.errInfoId(), "Room " + state.room.name() + " has
ended", "red");
        onStopClick(state);
    }).on(constants.SFU_ROOM_EVENT.DROPPED, function () {
        setStatus(state.errInfoId(), "Dropped from the room " +
state.room.name() + " due to network issues", "red");
        onStopClick(state);
    });
    startBitrateTest(state);
}
```

6. Тестирование канала

`SFURoom.join()`, `BitrateTest.setListener()`, `BitrateTest.test()` code

Функция `startBitrateTest()`:

- присоединяется к SFU комнате и настраивает WebRTC соединение
- получает доступ к объекту `BitrateTest`
- добавляет обработчик события `onStateChange` объекта `BitrateTest`
- запускает тест канала
- отображает результаты теста

```
const startBitrateTest = async function (state) {
    if (state.room) {
        await state.room.join(state.pc, null, {});
        const stateSelector = $("#" + state.currentStateId());
        stateSelector.attr("style", "display:inline-block;margin-left:
10px");
        try {
            const bitrateTest = state.room.getBitrateTest();
            state.setBitrateController(bitrateTest);
            bitrateTest.setListener({
                onStatusUpdate(bitrateKbps) {
                    stateSelector.text("Current bitrate: " + bitrateKbps + "
kbps");
                }
            });
            bitrateTest.test($("#" +
bitrateTestState.durationId()).val()).then((bitrateKbps) => {
                stateSelector.text("Test is finished, last measured bitrate:
" + bitrateKbps + " kbps");
                state.setBitrateController(null);
                onStopClick(state);
            });
```

```
        } catch (e) {
            if (e.type === constants.SFU_ROOM_EVENT.OPERATION_FAILED) {
                onOperationFailed(state, e);
            } else {
                console.error("Failed to start bitrate test: " + e);
                setStatus(state.errInfoId(), e.name, "red");
                onStopClick(state);
            }
        }
    }
}
```

7. Остановка теста

`BitrateTest.stop()` code

```
const stopBitrateTest = function (state) {
    const controller = state.getBitrateController();
    if (controller) {
        controller.stop();
    }
}
```

8. Действия по нажатию кнопки Start

`connect()` code

Функция `onStartClick()`:

- проверяет правильность заполнения полей ввода
- вызывает функцию `connect()`

```
const onStartClick = function (state) {
    if (validateForm("connectionForm", state.errInfoId())) {
        $("#" + state.buttonId()).prop('disabled', true);
        connect(state);
    }
}
```

9. Действия по нажатию кнопки Stop

`Session.disconnect()` code

Функция `onStopClick()`:

- останавливает тест, если он запущен
- разрывает Websocket сессию

```
const onStopClick = async function (state) {
    if (state.isConnected()) {
        stopBitrateTest(state);
        await state.session.disconnect();
        onDisconnected(state);
    }
}
```

10. Действия при разрыве Websocket сессии

code

Функция `onDisconnected()`:

- настраивает действия по нажатию кнопки `Start`
- открывает доступ к полям ввода `Server url` и `Test duration`

```
const onDisconnected = function (state) {
    state.clear();
    $("#" + state.buttonId()).text("Start").off('click').click(function () {
        onStartClick(state);
    }).prop('disabled', false);
    $('#url').prop('disabled', false);
    $("#" + bitrateTestState.durationId()).prop('disabled', false);
}
```