

Sistema di videosorveglianza basato su WebRTC

De Meo Claudio

Lionetti Valerio

Abstract - WebRTC è un framework open source basato su HTML5 e JavaScript che permette di realizzare comunicazioni real time peer to peer mediante browser o applicazioni mobili. Nel progetto realizzato si pone l'obiettivo di mostrare le potenzialità di WebRTC realizzando un sistema di videosorveglianza in cui viene implementato un algoritmo di motion detection.

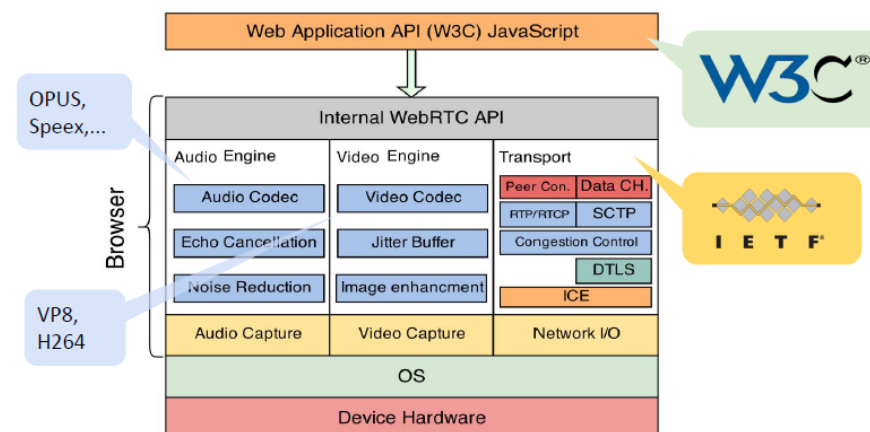
1 Introduzione - WebRTC

WebRTC - acronimo di Web Real Time Communication - è un framework per la comunicazione multimediale sul web standardizzato prima a livello API al W3C (World Wide Web Consortium) e poi a livello di protocollo presso l'IETF (Internet Engineering Task Force).



Con WebRTC, le aziende possono avere l'opportunità di trasformare le loro comunicazioni garantendo comunicazioni di classe enterprise affidabili e sicure. La sicurezza e la crittazione sono funzionalità nativamente integrate proprio a tutela della sicurezza. Inoltre WebRTC offre crittazione punto-punto su praticamente tutti i server, garantendo protezione e comunicazioni real-time sicure e private. WebRTC un progetto open source promosso da Google che permette senza l'installazione di software, plug-in o applicazioni particolari, comunicazioni in tempo

reale tramite le API di JavaScript utilizzando qualsiasi browser web. Per quanto riguarda il video i codec principalmente utilizzati sono VP8 o H264, mentre per l'audio vengono utilizzati principalmente OPUS o Speex.



Le API JavaScript utilizzate da WebRTC sono:

- **MediaStream:** detta anche `getUserMedia`, consente di ottenere, previo consenso dell'utente, un flusso audio/video proveniente dalla telecamera e dal microfono installato sul device dell'utente.
- **MediaRecorder:** permette di memorizzare video e audio provenienti dagli stream locali e remoti.
- **RTCPeerConnection:** consente di gestire la connessione e la comunicazione di flussi di dati tra due device in connessione peer-

to-peer, gestisce il workflow ICE per il NAT traversal. Utilizza un Server STUN (Simple Traversal of UDP Through NAT) che permette di scoprire l'IP pubblico di un peer che si trova dietro NAT. Mantiene traccia degli stream locali e remoti e usa SDP per la negoziazione e l'inizializzazione della connessione.

- RTCDataChannel: consente l'interscambio di flussi di dati arbitrari tra due device in connessione peer-to-peer.

1.1 Descrizione del sistema

Sfruttando la versatilità di WebRTC è stato realizzato, al fine di dimostrare le potenzialità del suddetto framework, un sistema di videosorveglianza in cui si hanno diverse webcam, che simulano le telecamere dell'impianto e acquisiscono mediante riprese video una scena da monitorare, e una stazione di controllo a cui arrivano, mediante peer connection, gli stream video dalle varie webcam. In oltre su ogni webcam è stato implementato un algoritmo di motion detection che, analizzando lo stream video frame per frame, rileva, in tempo reale, una regione all'interno del video interessata da movimento. Le informazioni così ottenute sul movimento vengono inviate alla stazione di controllo centrale attraverso un datachannel che ogni webcam crea con la stazione di controllo. La stazione di controllo, dunque, una volta ottenuti gli stream remoti dalle varie webcam e i dati sul movimento relativi ad ogni webcam, avrà il compito di mostrare all'utente le scene riprese dalle webcam evidenziando per ognuna di esse l'area in cui è stato rilevato il movimento, dovrà memorizzare i video ripresi e i dati relativi al movimento in un file di log che per ogni webcam sarà organizzato nel modo seguente:

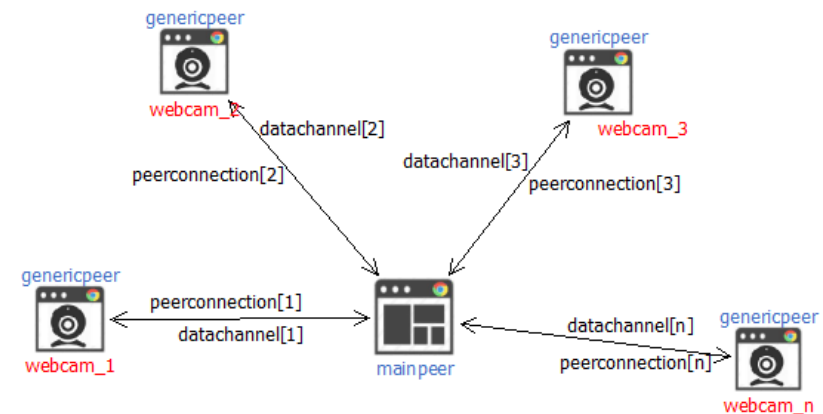
id_webcam; area_con_movimento; timestamp_del_movimento

2 Architettura

Il sistema è stato realizzato come un'applicazione web in cui sono state create due pagine web:

- una pagina, chiamata *genericpeer*, con il ruolo di chiamante, che permette all'utente di aggiungere o rimuovere la relativa webcam al sistema;
- una pagina, chiamata *mainpeer*, con il ruolo di chiamato, che permette all'utente di visualizzare tutti gli stream delle webcam collegate al sistema.

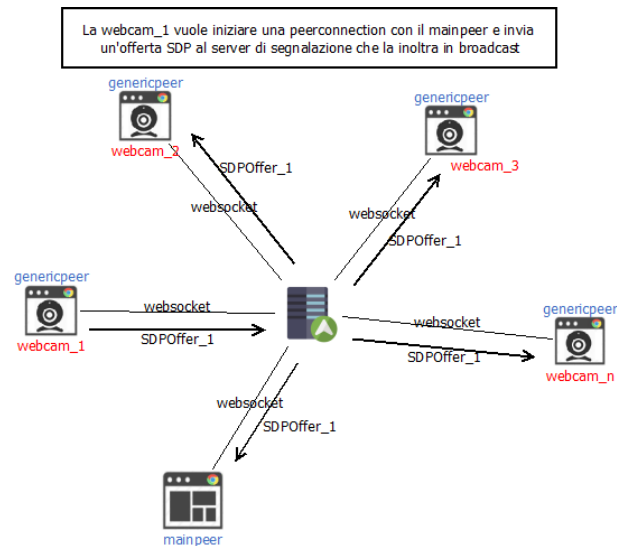
Tramite la pagina *genericpeer* ogni webcam effettua una chiamata sdp al *mainpeer* con il quale instaura una peerconnection e crea un datachannel con cui invia gli stream video delle webcam e le informazioni sul movimento.



Per instaurare le peerconnection, ogni *genericpeer* deve scambiare con il *mainpeer* il suo SDP. Lo scambio del pacchetto SDP viene gestito tramite un server di segnalazione al quale inizialmente si collegano tutti i nodi del sistema tramite una websocket.

2.1 Il server di segnalazione - Scambio SDP

Il nodo chiamante (in questo caso tramite la pagina *genericpeer*) invia il suo SDP al server di segnalazione che semplicemente inoltra in broadcast a tutti i nodi ad esso connessi il pacchetto SDP.

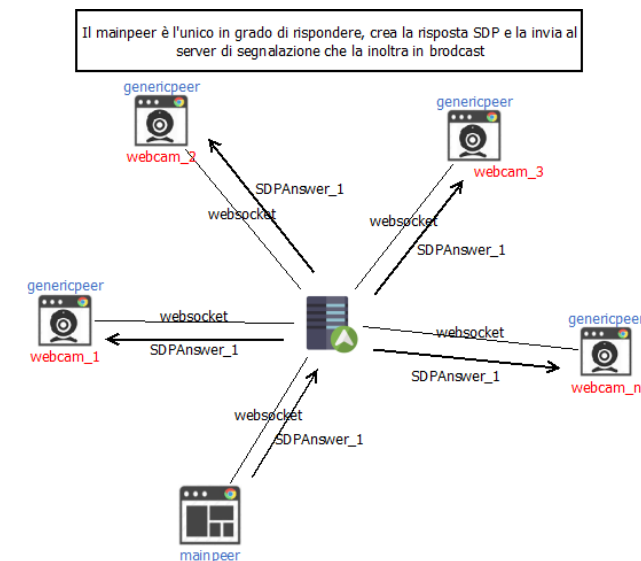


A questo punto il messaggio contenente l'SDP viene recapitato a tutti i nodi, ma l'unico in grado di interpretarlo è il nodo chiamato collegato al sistema tramite la pagina *mainpeer*, mentre eventuali altri nodi *genericpeer* scartano il messaggio in quanto un *genericpeer* non è abilitato a leggere un messaggio SDP se si trova nelle seguenti due condizioni:

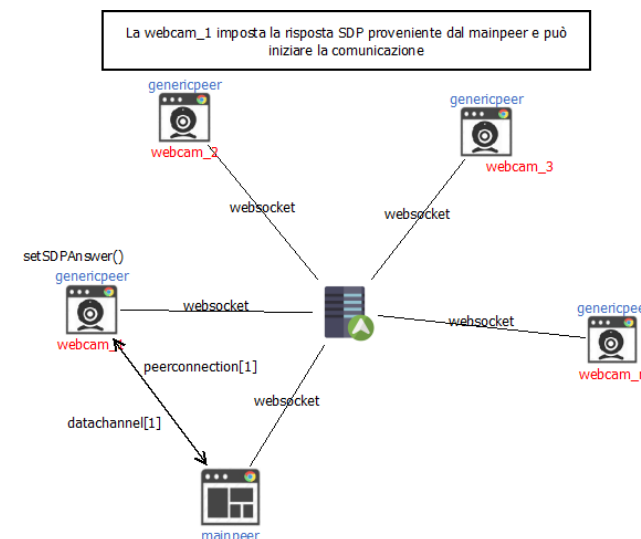
1. non ha effettuato una chiamata
2. ha completato uno scambio di SDP e ha instaurato una peerconnection.

In definitiva un *genericpeer* può leggere un SDP solo se ha effettuato una chiamata e attende l'SDP di risposta dal chiamato. Allo stesso modo la risposta del chiamato (tramite la pagina *mainpeer*) contenente il suo SDP

viene inviata al server di segnalazione che lo inoltra in broadcast a tutti i nodi ad esso collegati.



Infine il chiamante conclude la creazione della peerconnection impostando la sua SDP Answer.



E così via finché tutte le webcam non instaurano una peerconnection con il *mainpeer*.

Lo scambio di SDP tra chiamante e chiamato grazie al server di segnalazione avviene in maniera automatica. Per semplicità, non è stato implementato nessun meccanismo di sincronizzazione per l'accesso multiplo al server di segnalazione. Difatti, una webcam potrebbe iniziare uno scambio di SDP in qualunque momento. Esiste quindi il rischio che due o più webcam vogliano inviare un SDP contemporaneamente, o comunque prima che venga completata la precedente procedura di scambio di SDP tra due peer, ma considerando il numero limitato di webcam presenti nel sistema e essendo la procedura immediata e molto veloce si è assunto come ipotesi progettuale che la probabilità che questo avvenga sia molto bassa e questo problema è stato trascurato.

3. Motion Detection

Il rilevamento di movimento viene implementato confrontando pixel a pixel un frame con il successivo e se la differenza supera un certo valore di soglia si assume che il pixel sia variato significativamente e viene classificato come parte di un movimento. Il problema principale è la accuratezza, più grande è la risoluzione del frame, quindi più pixel vengono confrontati, e più è accurato l'algoritmo, ma è più costoso in termini di performance. La procedura di confronto tra tutti i pixel dell'immagine è di per sé molto lenta se si considera la risoluzione reale del frame, per questo motivo i frame vengono prima scalati ad una risoluzione di 64px x 48px e successivamente vengono confrontati:

```
function compare(image1, image2, width, height) {
    initialize();
    if(!image1 || !image2) {
        return;
    }
    temp1Context.clearRect(0,0,100000,100000);
    temp1Context.clearRect(0,0,100000,100000);
    temp1Context.drawImage(image1, 0, 0, width, height);
    temp2Context.drawImage(image2, 0, 0, width, height);
    for(var y = 0; y < height; y++) {
        for(var x = 0; x < width; x++) {
            var pixel1 = temp1Context.getImageData(x,y,1,1);
            var pixel1Data = pixel1.data;

            var pixel2 = temp2Context.getImageData(x,y,1,1);
            var pixel2Data = pixel2.data;

            if(comparePixel(pixel1Data, pixel2Data) == false) {
                setTopLeft(x,y);
                setBottomRight(x,y);
            }
        }
    }

    return {
        'topLeft': topLeft,
        'bottomRight': bottomRight
    }
}
```

Una volta ottenute la posizione e l'ampiezza del movimento è possibile ricalcolarla sull'immagine a più alta risoluzione. In questo modo viene calcolato un riquadro contenente il movimento in cui sono specificati i vertici in alto a destra e in basso a sinistra:

```
var vals = imageCompare.compare(currentImage, oldImage, width, height);
topLeft[0] = vals.topLeft[0] * 10;
topLeft[1] = vals.topLeft[1] * 10;
bottomRight[0] = vals.bottomRight[0] * 10;
bottomRight[1] = vals.bottomRight[1] * 10;
```

A questo punto i dati vengono rappresentati come coordinate del vertice in alto a sinistra, larghezza e altezza del riquadro e vengono inviati nel datachannel insieme all'id della webcam che riprende la scena attraverso la funzione `sendData()`:

```
var motion=document.getElementById("idcam").value+"\r\n top:"+topLeft[1] +
    "px;left:"+topLeft[0]+"px;width:"+((bottomRight[0] - topLeft[0])+"px;height:
    +(bottomRight[1] - topLeft[1])+"px;";
sendData(motion);

function sendData(data) {    //invia messaggi con data channel
    if(dataChannel!=null && connected){
        console.log("About to send this data: " + data);
        dataChannel.send(data);
    }
}
```

La sensibilità di questo algoritmo è data dunque dal valore di soglia prescelto:

```
function comparePixel(p1, p2) {
    var matches = true;
    var sensitivity=50;
    for(var i = 0; i < p1.length; i++) {
        var t1 = Math.round(p1[i]/10)*10;
        var t2 = Math.round(p2[i]/10)*10;
        if(t1 != t2) {
            if((t1+sensitivity < t2 || t1-sensitivity > t2)) {
                matches = false;
            }
        }
    }
    return matches;
}
```

Minore è il valore di soglia e minore è la variazione di pixel necessaria a rilevare un movimento, ma con un valore di soglia troppo basso potrebbero aumentare i casi di falsi positivi dovuti al rumore sempre presente della webcam.

4. Salvataggio dei video

Il sistema prevede il download di una registrazione di una camera specifica indicata dall'utente. Per permettere la registrazione si è utilizzata la classe *MediaRecorder*. La registrazione contemporanea di più camere è stata realizzata mediante l'uso di un array in cui ogni elemento contiene la registrazione della camera corrispondente. Il formato utilizzato per il download delle registrazioni è webm, con codec video vp9.

```
function startRecording(stream){
    recordedBlobs[i]= [];
    mediaRecorder[i] = new MediaRecorder(stream,{mimeType: 'video/webm; codecs=vp9'});
    console.log('Created MediaRecorder', mediaRecorder[i]);
    mediaRecorder[i].onstop = function (e){
        console.log('Recorder stopped: ', e);
    };
    mediaRecorder[i].ondataavailable=function(e){
        if (e.data && e.data.size > 0) {
            recordedBlobs[i].push(e.data);
        }
    };
    mediaRecorder[i].start();
    //mediaRecorder[i].requestData();
    console.log('MediaRecorder started', mediaRecorder[i]);
}
```

L'utente può scaricare in qualsiasi momento la registrazione della camera indicata attraverso la funzione *download*, attivata dal bottone corrispondente.

```
downloadVideoBtn.onclick=function (e){
    if (inputVideoText.value==""){
        alert("Inserire nome camera");
    }else{
        var trovato=false;
        var j=0;
        while (!trovato && j<=i){
            trovato=(movement[j].getAttribute("id")==inputVideoText.value);
            j++;
        }
    }
}
```



```

    if (trovato){
        mediaRecorder[j-1].requestData();
        setTimeout(function(){
            download(recordedBlobs[i],inputVideoText.value+'.webm','video/webm')
        },100);
    }else{
        alert("Camera "+inputVideoText.value+" non trovata");
    }
}

};

//funzione per il download dei dati in locale
function download(dato,nome,tipo){
    var blob = new Blob(dato, {type: tipo});
    var url = window.URL.createObjectURL(blob);
    var a = document.createElement('a');
    a.style.display = 'none';
    a.href = url;
    a.download = nome;
    document.body.appendChild(a);
    a.click();
    setTimeout(function() {
        document.body.removeChild(a);
        window.URL.revokeObjectURL(url);
    }, 100);
}

```

5. File di Log per il movimento

I dati relativi al movimento una volta arrivati al mainpeer vengono ordinati per webcam di provenienza e memorizzati in un array in cui ogni elemento corrisponde ai dati relativi ad una webcam. L'utente in qualsiasi momento potrà decidere di scaricare i dati i quali verranno riordinati in formato JSON e successivamente scaricati attraverso la funzione download():

```

downloadlogBtn.onclick=function (e){
    var tempLog="{\r\n";
    for (j=0;j<=i;j++){
        tempLog=tempLog+log[j]+", "+'\r\n';
    }
    tempLog=tempLog.substring(0,tempLog.length-3)+"\r\n";
    //download dei file di log
    download([tempLog],"log.json","application/octet-stream");
};

//funzione per il download dei dati in locale
function download(dato,nome,tipo){
    var blob = new Blob(dato, {type: tipo});
    var url = window.URL.createObjectURL(blob);
    var a = document.createElement('a');
    a.style.display = 'none';
    a.href = url;
    a.download = nome;
    document.body.appendChild(a);
    a.click();
    setTimeout(function() {
        document.body.removeChild(a);
        window.URL.revokeObjectURL(url);
    }, 100);
}

```

La struttura del file JSON è la seguente:

```

{
  "idPeer": [{
    "durata": "intervallo di tempo in ms",
    "movimento": [{
      "box": "coordinate e ampiezza del movimento",
      "time": "timestamp del movimento"
    }]
  }]
}

```

Di seguito è mostrato un estratto di file di log con movimenti relativi a due peer:

```
"idPeer1":[{
  "durata": "115ms" ,
  "movimento": [{
    "box": "top:270px;left:490px;width:0px;height:0px;",
    "time": "Thu Apr 27 2017 22:40:54 GMT+0200 (ora legale Europa occidentale)"
  },
  {
    "box": "top:Infinitypx;left:Infinitypx;width:-Infinitypx;height:-Infinitypx;",
    "time": "Thu Apr 27 2017 22:40:54 GMT+0200 (ora legale Europa occidentale)"
  },
  ]
},
{
  "durata": "113ms" ,
  "movimento": [{
    "box": "top:270px;left:320px;width:170px;height:170px;",
    "time": "Thu Apr 27 2017 22:40:54 GMT+0200 (ora legale Europa occidentale)"
  },
  {
    "box": "top:Infinitypx;left:Infinitypx;width:-Infinitypx;height:-Infinitypx;",
    "time": "Thu Apr 27 2017 22:40:55 GMT+0200 (ora legale Europa occidentale)"
  },
  ]
},
]
},
{idPeer2":[{
  "durata": "85ms" ,
  "movimento": [{
    "box": "top:340px;left:370px;width:0px;height:0px;",
    "time": "Thu Apr 27 2017 22:40:56 GMT+0200 (ora legale Europa occidentale)"
  },
  ]
},
]
```

Come si può notare sono presenti alcune box con i campi “top”, “left”, “width” e “height” impostati a “Infinity”. Questi valori rappresentano il momento in cui è cessato un movimento. Si è scelto di rappresentare anche questi casi in modo da permettere all’utente di individuare la fine di un movimento.

6. Utilizzo del sistema

Di seguito vengono descritte le modalità d’uso del sistema realizzato.

6.1. Avvio server di segnalazione

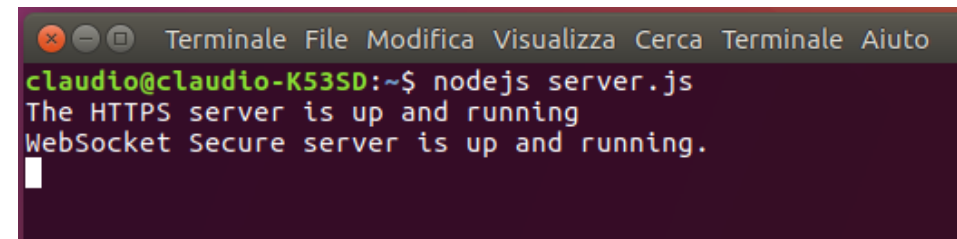
La prima cosa da fare per utilizzare il sistema è avviare il server di segnalazione affinché i peer possano scambiarsi i pacchetti SDP. Esso viene avviato tramite linea di comando posizionandosi nella directory che lo contiene e lanciando il seguente comando:

- Per macchine linux:

nodejs server.js

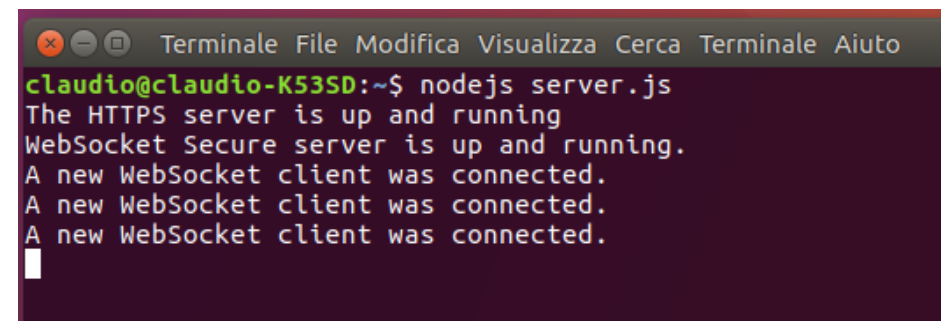
- Per macchine windows:

node server.js



```
Terminale File Modifica Visualizza Cerca Terminale Aiuto
claudio@claudio-K53SD:~$ nodejs server.js
The HTTPS server is up and running
WebSocket Secure server is up and running.
```

Quando una pagina web viene aperta per prima cosa si collega al server di segnalazione:



```
Terminale File Modifica Visualizza Cerca Terminale Aiuto
claudio@claudio-K53SD:~$ nodejs server.js
The HTTPS server is up and running
WebSocket Secure server is up and running.
A new WebSocket client was connected.
A new WebSocket client was connected.
A new WebSocket client was connected.
```

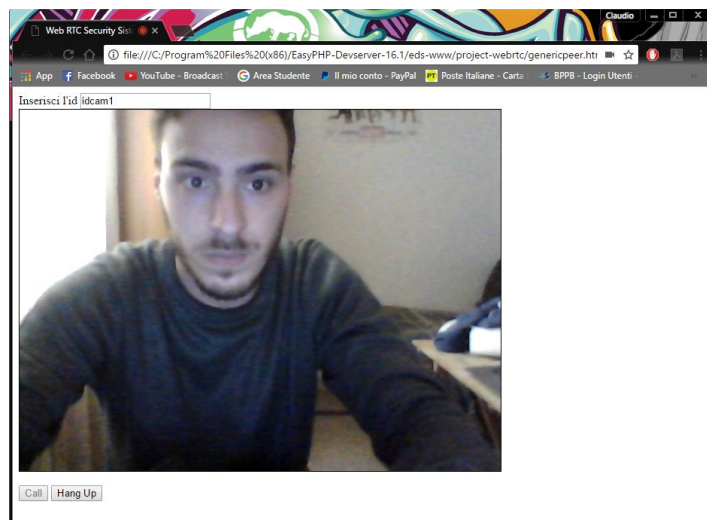
Il server di segnalazione notifica quando un nuovo client viene aggiunto.

A questo punto un genericpeer può inviare il suo SDP al server di segnalazione, che a sua volta invia in broadcast a tutti i peer ad esso collegati mostrando il seguente messaggio:

```
Terminale File Modifica Visualizza Cerca Terminale Aiuto
claudio@claudio-K53SD:~$ nodejs server.js
The HTTPS server is up and running
WebSocket Secure server is up and running.
A new WebSocket client was connected.
A new WebSocket client was connected.
A new WebSocket client was connected.
Broadcasting message to all 3 WebSocket clients.
Broadcasting message to all 3 WebSocket clients.
Broadcasting message to all 3 WebSocket clients.
Broadcasting message to all 3 WebSocket clients.
```

6.2. Inizializzazione delle camere

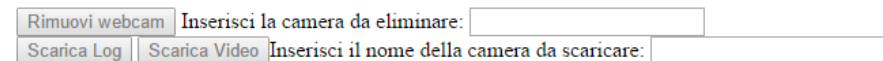
L'utente accede alla pagina web genericpeer.html, dove inserisce l'id della camera ed invia l'offerta sdp al server di segnalazione con il bottone Call, dove verrà instaurata la peerconnection automaticamente con la pagina di monitoraggio mainpeer.html.



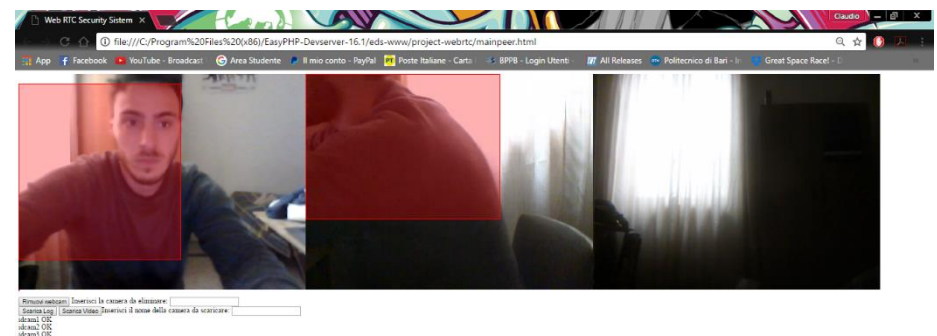
Vi è la possibilità di chiudere lato camera una connessione mediante l'uso del bottone Hung Up.

6.3. Gestione pagina monitoraggio

La pagina web adibita al monitoraggio globale delle camere connesse è mainpeer.html e si presenta inizialmente nel seguente modo:



Inizialmente i pulsanti sono disabilitati e l'utente non può compiere nessun'azione finché un genericpeer non instaura una peerconnection. Nel momento in cui si instaura una peerconnection sarà dinamicamente aggiunto e visualizzato lo stream della relativa camera. La presenza del movimento sarà evidenziata, mediante delle box, nel video corrispondente.



L'utente può rimuovere le camere inserendo l'id corrispondente, scaricare il file di log generale (contenente i movimenti di tutte le eventuali

camere connesse) e scaricare la registrazione video di una camera tramite id.

7. Conclusioni e sviluppi futuri

In conclusione il progetto realizzato ha messo in evidenza con come WebRTC si presta bene ad applicazioni di sicurezza come la videosorveglianza, in quanto dispone di una serie di funzionalità native proprio a tutela della sicurezza. WebRTC richiede che l'utente autorizzi esplicitamente l'uso della sua webcam e microfono. Ciò garantisce che l'utente sia a conoscenza di quando questi dispositivi siano accesi. Per trasferire dati con WebRTC in tempo reale, essi vengono prima crittati utilizzando il metodo DTLS (Datagram Transport Layer Security). Questo è un protocollo integrato fin dall'origine in tutti i browser che supportano WebRTC (Chrome, Firefox e Opera). Oltre al metodo DTLS, WebRTC critta audio e video con il metodo SRTP (Secure Real-Time Protocol), garantendo che le comunicazioni IP non possano essere nè viste nè sentite da soggetti non autorizzati. WebRTC offre quindi crittazione punto-punto su praticamente tutti i server, garantendo protezione e comunicazioni real-time sicure e private.

Per sfruttare a pieno le potenzialità di WebRTC in futuro si può pensare di implementare un server MCU (Multipoint Conferencing Unit) come Kurento in alternativa al server di segnalazione utilizzato in questo progetto, per gestire in maniera automatica ed efficiente sia lo scambio dei pacchetti SDP che il problema dell'accesso multiplo. Inoltre è possibile realizzare un sistema di notifiche che, sulla base del file di log generato, avvisi, in real time, l'utente amministratore del sistema della presenza di un movimento in una scena ripresa.

Riferimenti esterni

- <https://webrtc.org/>
- <https://www.w3.org/TR/webrtc/>
- <https://codelabs.developers.google.com/codelabs/webrtc-web/#0>
- <https://github.com/webrtc/samples>