

ETRS 902 : SIP et Asterisk avancés

Séance 1 - Déploiement D'Asterisk	2
1.1 Identifiez les fichiers de configurations	2
Séance 2 - Trunk SIP et NAT	3
2.2 Faire un appel de Spub à Spriv de ne passe pas car le NAT bloque la transmission	3
2.3 Etablissement des appels, solution « stun »	3
2.4 Etablissement des appels, solution « rport »	4
Séance 3 - Stockage persistant et CDR dans MySQL	5
3.1 Gestion des CDR dans MySQL	5
3.2 Gestion de astDB à partir du « Dialplan »	7
Séance 4 – Développement d'applications externes	8
4.1 Activez et Vérifier AMI	8
4.2 Activer ARI pour l'appel des REST API	10
4.3 Mettre en place l'environnement Web	10
4.4 Lister des téléphones	11
4.5 Clic 2 Call	12
Séance 5 - Menu vocal et synthèse vocale	14
5.1 Fichiers créés	14
5.2 tp4.json	14
5.3 pharserJson.php	15
5.4 extensions.conf	16
Séance 6 - ARI et Websocket	16
6.1 WebSocket Implémentation et Socket.io	16
6.2 Schéma	16
6.3 Mettre en place	17
6.4 Résultat	19

I. Séance 1 - Déploiement D'Asterisk

1.1 Identifiez les fichiers de configurations

Le fichier **pjsip.conf** nous sert à déclarer les hôtes qui vont interagir avec l'Asterisk pour communiquer (téléphones , passerelle, autres serveurs...). Les hôtes ont un extension par défaut.

Le fichier **extensions.conf** permet de savoir quels numéros vont effectuer des actions définies pour chaque numéros et savoir le déroulement de l'appel.

Le fichier **cdr_mysql.conf** permet de configurer une base de données SQL en local.

Le fichier **manager.conf** indique les programmes externes pouvant communiquer avec le serveur Asterisk

Le fichier **voicemail.conf** instancie les différentes boîtes vocales ainsi que leurs droits.

II. Séance 2 - Trunk SIP et NAT

2.2 Faire un appel de Spub à Spriv de ne passe pas car le NAT bloque la transmission

1) Test entre entre Spub dans TP_VOIP0 et Spriv dans TP_VOIP1

Appel Spub > Spriv	Appel Spriv > Spub
Le téléphone Spub déclenche la sonnerie mais le téléphone Spriv ne reçoit pas la notification Ringing	Le téléphone Spriv appelle le téléphone Spub qui décroche en recevant le Ringing. Le flux RTP est asymétrique. Seul le flux RTP allant de Spriv à Spub passe. Quand Spub raccroche en premier, Spriv reste en ligne

2) Test entre entre Spub dans TP_VOIP0 et Spriv dans TP_VOIP2

Appel Spub > Spriv	Appel Spriv > Spub
Le téléphone Spub déclenche la sonnerie mais le téléphone Spriv ne reçoit pas la notification Ringing	Le téléphone Spriv appelle le téléphone Spub qui décroche en recevant le Ringing. Le flux RTP est asymétrique. Seul le flux RTP allant de Spriv à Spub passe. Quand Spub raccroche en premier, Spriv reste en ligne

2.3 Etablissement des appels, solution « stun »

Juste après avoir configuré le serveur STUN, l'appel de Spub vers Spriv se passe correctement avec un flux RTP symétrique.

Après avoir attendu 3 minutes et que Spriv a relancé son application, l'appel ne passe pas car l'entrée NAT du routeur n'existe plus

Pour palier à ça, il faut un keep-alive

1) Test entre entre Spub dans TP_VOIP0 et Spriv dans TP_VOIP1 avec serveur STUN

Appel Spub > Spriv	Appel Spriv > Spub
--------------------	--------------------

<p>Le téléphone Spub déclenche la sonnerie mais le téléphone Spriv ne reçoit pas la notification Ringing.</p> <p>Avec un serveur STUN, la communication peut s'établir entre les deux téléphones</p> <p>Après 3 minutes, la communication n'est plus possible</p>	<p>Le téléphone Spriv appelle le téléphone Spub qui décroche en recevant le Ringing. Le flux RTP est symétrique. Quand l'un raccroche en premier, l'autre raccroche aussi</p>
---	---

2) Test entre Spub dans TP_VOIP0 et Spriv dans TP_VOIP2 avec serveur STUN

Appel Spub > Spriv	Appel Spriv > Spub
<p>Le téléphone Spub déclenche la sonnerie mais le téléphone Spriv ne reçoit pas la notification Ringing</p>	<p>Le téléphone Spriv appelle le téléphone Spub qui décroche en recevant le Ringing. Le flux RTP est asymétrique. Seul le flux RTP allant de Spriv à Spub passe. Quand Spub raccroche en premier, Spriv reste en ligne</p>

2.4 Etablissement des appels, solution « rport »

Le téléphone Spriv appelle le téléphone Spub qui décroche en recevant le Ringing. Le flux RTP est symétrique. Quand Spub raccroche en premier, Spriv raccroche aussi et vice versa

III. Séance 3 - Stockage persistant et CDR dans MySQL

3.1 Gestion des CDR dans MySQL

- 1) Installer mysql client et mysql server
- 2) Créez une base de données CDR et un utilisateur Asterisk
 - a. pour accéder la console MySQL:
mysql -u root -puser
 - b. créer une base de données "asterisk":
create database asterisk;
 - c. créer un compte "asterisk":
create user 'asterisk'@'localhost' IDENTIFIED BY 'root';
 - d. donner la permission complète au compte "asterisk":
grant all on asterisk.* to 'asterisk'@'localhost';
- 3) Créez la table de stockage des CDR
 - a. créer un fichier tableCdr.sql avec le contenu ci-dessous:

```
CREATE TABLE cdr (  
    calldate datetime NOT NULL default '0000-00-00 00:00:00',  
    clid varchar(80) NOT NULL default '',  
    src varchar(80) NOT NULL default '',  
    dst varchar(80) NOT NULL default '',  
    dcontext varchar(80) NOT NULL default '',  
    channel varchar(80) NOT NULL default '',  
    dstchannel varchar(80) NOT NULL default '',  
    lastapp varchar(80) NOT NULL default '',  
    lastdata varchar(80) NOT NULL default '',  
    duration int(11) NOT NULL default '0',  
    billsec int(11) NOT NULL default '0',  
    disposition varchar(45) NOT NULL default '',  
    amaflags int(11) NOT NULL default '0',  
    accountcode varchar(20) NOT NULL default '',  
    uniqueid varchar(32) NOT NULL default '',  
    userfield varchar(255) NOT NULL default '',  
    peeraccount varchar(20) NOT NULL default '',
```

```
linkedid varchar(32) NOT NULL default '',  
sequence int(11) NOT NULL default '0');
```

b. lancer la commande:

```
mysql -u root -puser asterisk < /home/user/tableCdr.sql
```

- 4) Configurez « cdr_mysql.conf ». Testez à l'aide de la CLI que le backend de gestion des CDR a bien été pris en compte.

voir le contenu du cdr_mysql.conf:

```
root@debian:/home/user# cat /etc/asterisk/cdr_mysql.conf  
[global]  
hostname=127.0.0.1  
dbname=asterisk  
table=cdr  
password=root  
user=asterisk  
port=3306
```

service asterisk restart

asterisk -rvvv

cdr show status

```
* Registered Backends  
-----  
mysql  
Adaptive ODBC  
cdr-custom  
csv  
cdr_manager (suspended)  
res_config_sqlite
```

donc cdr-mysql a bien été pris en compte.

- 5) Faites quelques appels, vérifier que les CDR sont bien stockés (SELECT * FROM cdr ...) à partir de la console MySQL

mysql -u root -puser

USE asterisk

**SELECT calldate AS date , src AS appelant, dst AS appelé, duration AS durée
FROM cdr;**

(Pour vider le tableau "cdr": **TRUNCATE TABLE `cdr`;**)

```
MariaDB [asterisk]> SELECT calldate AS date , src AS appelant, dst AS appelé, duration AS durée FROM cdr;  
+-----+-----+-----+-----+  
| date           | appelant | appelé | durée |  
+-----+-----+-----+-----+  
| 2019-12-18 00:32:33 | 5860     | 101    | 8      |  
| 2019-12-19 08:31:27 | 5860     | 07719  | 2      |  
| 2019-12-19 08:31:45 | 5860     | 07719  | 8      |  
| 2019-12-19 08:33:05 | 5860     | 08738  | 55     |  
+-----+-----+-----+-----+
```

3.2 Gestion de astDB à partir du « Dialplan »

- 1) Mise en place du contexte pour ajouter/supprimer la redirection:

Une variable s'initialise en récupérant la valeur d'une clé présente ou non dans la base de données. Si la valeur est NULL alors on lui demande un numéro interne pour le renvoi d'appel sinon on supprime le renvoi

```
[renvoi]

exten=> 88888,1,Set(VERIF=${DB(renvoi/${CALLERID(num)})})
same=> n,NoOp(${VERIF})
same=> n,GotoIf(${ISNULL(${VERIF})})?:effacer)
same=> n,Read(digits,,2)
same=> n,NoOp(${digits})
same=> n,Set(DB(renvoi/${CALLERID(num)})${digits})
same=> n,Hangup()
same=> n(effacer),NoOp(${DB_DELETE(renvoi/${CALLERID(num)})})
same=> n,Hangup()
```

Vérifier que notre entrée est bien dans la base de données (database show)

```
debian*CLI> database show
/dundi/secret : 2CzHFm2jbkVQxEVeKr2GMw==;BkcXQBtRowBEjU82oMttAw==
/dundi/secretexpiry : 1576761438
/pbx/UUID : b288d0b5-8cfd-45cc-bb05-d2c595a7fdda
/registrar/contact/5861;@19c1660c9cc5ffe140ead843c928dda4: {"qualify_timeout":"3.000000","reg_server":"","cal
"192.168.141.181","outbound_proxy":"","expiration_time":"1576761233","path":"","endpoint":"5861","qualify_fr
,"authenticate_qualify":"no","uri":"sip:5861@192.168.141.181:64962;transport=UDP;rinstance=8791af53ac8553ef",
/renvoi/5861 : 60
5 results found.
```

Test de vérification lors d'un appel en interne

```
[appels-interne]

exten=> _6[0-1],1,Set(VERIF=${DB(renvoi/58${EXTEN})})
same=> n,GotoIf(${ISNULL(${VERIF})})?:renvoi)
same=> n,Dial(PJSIP/58${EXTEN},20)
same=> n,VoiceMail(600${EXTEN:1}@voicemail,u)
same=> n(renvoi),Dial(PJSIP/58${VERIF},20)
same=> n,VoiceMail(600${EXTEN:1}@voicemail,u)
exten=> 999,1,Goto(IVR,s,1)
```

IV. Séance 4 – Développement d'applications externes

4.1 Activez et Vérifier AMI

- 1) éditer le fichier /etc/asterisk/manager.conf

login: toto

mot de passe: pass1

```
; By default asterisk will listen on localhost only.
[general]
enabled = yes
port = 5038
bindaddr = 0.0.0.0
webenabled = yes
enabled = yes

; Each user has a section labeled with the username
; so this is the section for the user named "toto"
[toto]
secret = pass1
permit = 0.0.0.0/0.0.0.0
read = system,call,log,verbose,command,agent,user,originate
write = system,call,log,verbose,command,agent,user,originate
```

- 2) vérifier le port 5038 est ouvert

```
root@debian:/home/user# lsof -iTCP -sTCP:LISTEN
COMMAND  PID    USER  FD   TYPE DEVICE SIZE/OFF NODE NAME
vsftpd   377    root   3u   IPv6 12817 0t0     TCP *:ftp (LISTEN)
sshd     398    root   3u   IPv4 13480 0t0     TCP *:ssh (LISTEN)
sshd     398    root   4u   IPv6 13482 0t0     TCP *:ssh (LISTEN)
mysqld   503    mysql  19u  IPv4 13847 0t0     TCP localhost:mysql (LISTEN)
asterisk 1197   asterisk 7u   IPv4 18550 0t0     TCP *:5038 (LISTEN)
asterisk 1197   asterisk 18u  IPv4 18700 0t0     TCP *:cisco-sccp (LISTEN)
```


- 3) désactiver le firewall sur client (windows)
- 4) tester AMI via Telnet

```
Asterisk Call Manager/2.9.0
Action: login
Username: toto
Secret: pass1

Response: Error
Message: Missing action in request

Action: login
Username: toto
Secret: pass1

Response: Success
Message: Authentication accepted

Event: FullyBooted
Privilege: system,all
Status: Fully Booted

Event: SuccessfulAuth
Privilege: security,all
EventTV: 2019-12-18T23:15:24.682+0100
Severity: Informational
Service: AMI
```

- 5) tester AMI over HTTP

<http://192.168.141.235:8088/manager?action=login&username=toto&secret=pass1>

Not secure | 192.168.141.235:8088/manager?action=login&username=toto&secret=pass1 ☆

aidu Sac à bandoulière... 1分钟搭建自定义域... How to Set Up HT... Google Hack技巧... Custom Search JS... 14 Free VPN

Manager Tester

Action: or

CLI Command

user pass

Response	Success
Message	Authentication accepted

6) lister les endpoints via la page de manager du serveur asterisk

Not secure | 192.168.141.235:8088/manager

Sac à bandoulière... 1分钟搭建自定义域... How to Set Up HT... Google Hack技巧... Custom Search JS... 14 Free VPN With...

Manager Tester

Action: <-----> or <----->
CLI Command <----->
user <-----> pass <----->
Submit

Response Follows
Privilege Command

Endpoint <Endpoint/CID.....> <State.....> <Channels.>
I/OAuth <AuthId/UserName.....>
Aor <Aor.....> <MaxContact>
Contact <Aor/ContactUri.....> <Hash.....> <Status> <RTT(ms).....>
Transport <TransportId.....> <Type> <cos> <tos> <BindAddress.....>
Identify <Identify/Endpoint.....>
Match <ip/cidr.....>
Channel <ChannelId.....> <State.....> <Time.....>
Exten <DialedExten.....> CLCID: <ConnectedLineCID.....>

Endpoint: 5860 Unavailable 0 of inf
InAuth: 5860/user1
Aor: 5860 3
Endpoint: 5861 Not in use 0 of inf
InAuth: 5861/5861
Aor: 5861 3
Contact: 5861/sip:5861@192.168.141.181:64962;transp 19c1660c9c Unknown nan

Opaque-
data

4.2 Activer ARI pour l'appel des REST API

1) Editer /etc/asterisk/ari.conf

login: toto

mot de passe: pass1

```
[general]
enabled = yes
pretty = yes
allowed_origins = *

[toto]
type = user
read_only = no
password = pass1
```

2) Tester Rest API pour récupérer la liste de endpoints

curl -v -u toto:pass1 -X GET "http://192.168.141.235:8088/ari/endpoints"

4.3 Mettre en place l'environnement Web

1) Activer Serveur HTTP interne dans Asterisk

Le Serveur Asterisk fournit un mini serveur HTTP, pour l'activer on configure /etc/asterisk/http.conf

```
[general]
enabled=yes
bindaddr=0.0.0.0
bindport=8088
```

Ensuite, on crée des pages HTML et scripts dans /usr/share/asterisk/static-http/

2) Installation du PHP

```
sudo apt-get install software-properties-common
sudo add-apt-repository ppa:ondrej/php
sudo apt-get update
sudo apt-get upgrade
sudo apt-get install -y php7.3
sudo apt-get install -y php7.3-cli php7.3-common php7.3-curl php7.3-mbstring
php7.3-mysql php7.3-json
```

4.4 Lister des téléphones

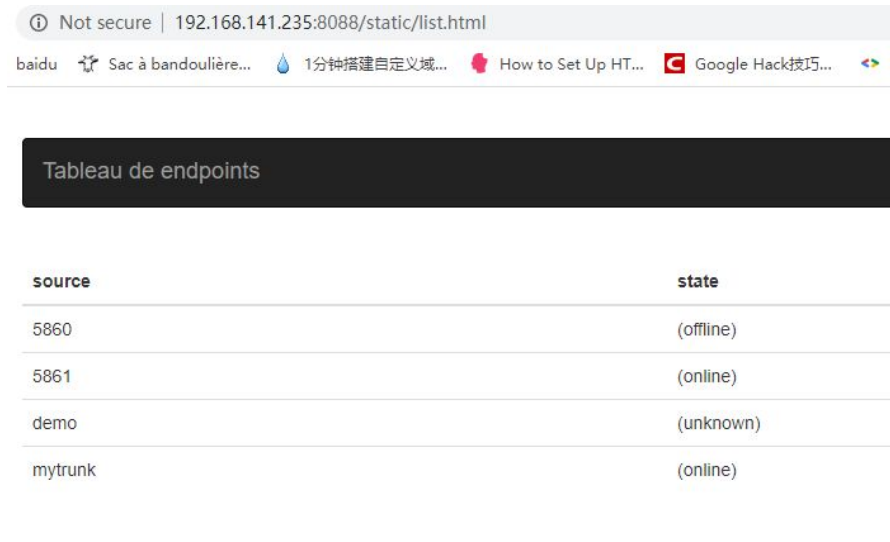
(Fichiers créés: **list.html**, **list.js** dans /usr/share/asterisk/static-http/)

On utilise ARI (Asterisk REST Interface) pour lister les endpoints on utilise l'api REST:
http://ip_serveur:8088/ari/endpoints via la méthode **GET**.

- 1) il faut authentifier un compte qui est défini dans le fichier ari.conf, dans ce cas on a défini un compte **<toto>** avec son mot de passe **<pass1>**.
- 2) pour l'authentification, ajouter un header dans une requête HTTP, dans ce cas on utilise Ajax:

```
function list_endpoints() {
$.ajax({
  type : "GET",
  url : "http://192.168.141.235:8088/ari/endpoints",
  dataType : 'json',
  headers : {
    "Authorization" : "Basic " + btoa("toto:pass1")
  },
  success : function(data) {
```

3) tester dans la page list.html



The screenshot shows a web browser with the address bar displaying "192.168.141.235:8088/static/list.html". The page title is "Tableau de endpoints". Below the title is a table with two columns: "source" and "state". The table contains four rows of data.

source	state
5860	(offline)
5861	(online)
demo	(unknown)
mytrunk	(online)

4.5 Clic 2 Call

(Fichiers créés: [cllic2call.html](#), [cllic2call.js](#) dans /usr/share/asterisk/static-http/)

On utilise l'api REST **http://ip_serveur:8088/ari/channels** via la méthode **POST**.

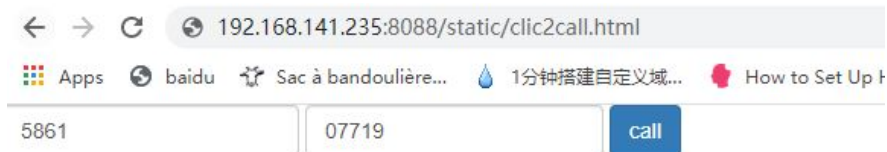
- 1) ajouter un header dans une requête HTTP et mettre des paramètres: endpoint, extension, context, priority et callerId, dans ce cas on utilise Ajax:

```
call();  
|  
function call() {  
  $.ajax({  
    type : "POST",  
    url : "http://192.168.141.235:8088/ari/channels",  
    dataType : 'json',  
    headers : {  
      "Authorization" : "Basic " + btoa("toto:pass1")  
    },  
    data : {  
      endpoint : "PJSIP/" + caller,  
      extension : called,  
      context : "internes",  
      priority : "1",  
      callerId : caller  
    },  
    success : function(res, textStatus, xhr) {  
      alert(xhr.status);  
    }  
  })  
}
```

2) tester dans la page clic2call.html et voir le résultat dans la console du serveur asterisk:

appellant: 5861

appelé: 07719



```
Connected to Asterisk 13.14.1~dfsg-2+deb9u4 currently running on debian (pid = 378)
-- Called 5861
-- PJSIP/5861-0000000f is ringing
> 0x1f1bcb8 -- Strict RTP learning after remote address set to: 192.168.141.181:65194
-- PJSIP/5861-0000000f answered
-- Executing [07719@internes:1] Dial("PJSIP/5861-0000000f", "PJSIP/7719@mytrunk,20") in new stack
-- Called PJSIP/7719@mytrunk
> 0x1f1bcb8 -- Strict RTP switching to RTP target address 192.168.141.181:65194 as source
-- PJSIP/mytrunk-00000010 is ringing
> 0x1f1bcb8 -- Strict RTP learning complete - Locking on source address 192.168.141.181:65194
> 0x1f16840 -- Strict RTP learning after remote address set to: 192.168.176.59:18000
-- PJSIP/mytrunk-00000010 answered PJSIP/5861-0000000f
-- Channel PJSIP/mytrunk-00000010 joined 'simple_bridge' basic-bridge <a97979de-2da7-4f58-af34-dce9e3b3610d>
-- Channel PJSIP/5861-0000000f joined 'simple_bridge' basic-bridge <a97979de-2da7-4f58-af34-dce9e3b3610d>
> Bridge a97979de-2da7-4f58-af34-dce9e3b3610d: switching from simple_bridge technology to native_rtp
> Locally RTP bridged 'PJSIP/5861-0000000f' and 'PJSIP/mytrunk-00000010' in stack
> 0x1f16840 -- Strict RTP switching to RTP target address 192.168.176.59:18000 as source
-- Channel PJSIP/mytrunk-00000010 left 'native_rtp' basic-bridge <a97979de-2da7-4f58-af34-dce9e3b3610d>
-- Channel PJSIP/5861-0000000f left 'native_rtp' basic-bridge <a97979de-2da7-4f58-af34-dce9e3b3610d>
== Spawn extension (internes, 07719, 1) exited non-zero on 'PJSIP/5861-0000000f'
```

V. Séance 5 - Menu vocal et synthèse vocale

5.1 Fichiers créés

- 1) Dans /usr/share/asterisk/static-http:
 - parserJson.php
 - tp4.json
- 2) Dans /usr/share/asterisk/sounds/
 - accueil_note.ulaw
 - dire_note.ulaw
 - assez_bien.ulaw
 - bien.ulaw
 - tres_bien.ulaw
 - echec.wav
 - victory.wav

5.2 tp4.json

Dans ce fichier json on a des fausses données ci-dessous pour tester:

```
[
  {
    "matricule": "111",
    "note": 15,
    "mention": "bien"
  },
  {
    "matricule": "222",
    "note": 11,
    "mention": "assez_bien"
  },
  {
    "matricule": "333",
    "note": 8,
    "mention": null
  }
]
```

5.3 pharserJson.php

```
<?php
// récupérer les variables dans la CLI
$matricule=$argv[1];
$action = $argv[2]; // action c'est note ou mention

// ouvrir le fichier tp4.json et décoder ces données en JSON
$file= file_get_contents('/usr/share/asterisk/static-http/tp4.json',
true);
$arr= json_decode($file, true);

// parcourir le tableau $arr
foreach ($arr as $element) {
    // comparer $action:
    //      - si $action est "note", afficher la note du matricule
    //      choisi ($matricule)
    //      - si $action est "mention", afficher la mention du
    //      matricule choisi ($matricule)
    //      - sinon afficher "action or note not found"
    switch($action){
        case "note":
            if($element['matricule']==$matricule){
                echo $element['note'];
            }
            break;
        case "mention":
            if($element['matricule']==$matricule){
                echo $element['mention'];
            }
            break;
        default:
            echo "action or note not found";
    }
}
?>
```

5.4 extensions.conf

```
[notesetudiants]; Séance 5 , outils permettant de connaître la note et mention d'un etudiant avec son matricule
;fichiers audios sont : accueil_note, dire_note, passable, assez_bien, bien, tres_bien, echec
exten => 514,1,Answer()
same => n,Playback(acceuil_note)
same => n,Read(MATRICULE,,3); on lit le matricule de l'etudiant
same => n,Set(NOTE=${SHELL(/usr/bin/php /usr/share/asterisk/static-http/pharserverJson.php ${MATRICULE} note)}); on recupere la note de l'etudiant
same => n,Set(MENTION=${SHELL(/usr/bin/php /usr/share/asterisk/static-http/pharserverJson.php ${MATRICULE} mention)}); on recupere la mention de l'étudiant
same=> n,NoOp(${NOTE})
same=> n,NoOp(${MENTION})
same => n,Playback(dire_note)
same => n,SayNumber(${NOTE}); on lit la note obtenue
same => n,GotoIf(${ISNULL(${MENTION}})?echec:oui) ; Si y a mention alors on va dans oui sinon on va dans echec
same => n(oui),Playback(${MENTION})
same => n,Playback(victory)
same => n,Hangup()
same => n(echec),Playback(echec)
same => n,Hangup()
```

Le contexte ci-dessous permet de faire le lien entre un fichier PHP et un matricule demandé par un élève. On demande d'abord à l'utilisateur de rentrer le matricule. Une fois le matricule rentré, on exécute 2 fois un script pour récupérer la note dans une variable et la mention dans une autre. On vérifie si l'utilisateur a sa note et on vérifie ensuite sa mention pour lui dire.

VI. Séance 6 - ARI et Websocket

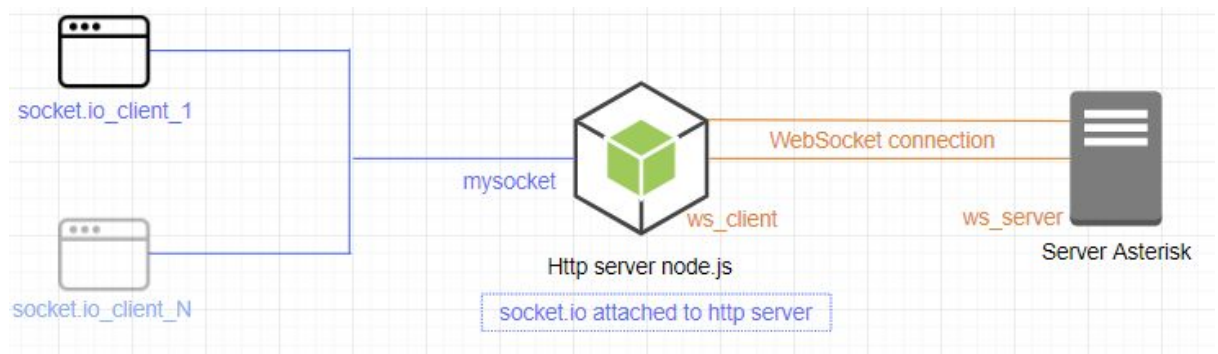
6.1 WebSocket Implémentation et Socket.io

- WebSocket
il est un protocole qui est établi sur HTTP. C'est un protocole de connection persistant, il propose une implémentation native pour créer une WebSocket connection via l'url suivant ws://xxxxx ou wss://xxxxx (comme http ou https).
- Socket.IO
elle est une bibliothèque JavaScript pour les applications Web en temps réel. Il permet une communication bidirectionnelle en temps réel entre les clients Web et les serveurs. Il comporte deux parties: une bibliothèque côté client qui s'exécute dans le navigateur et une bibliothèque côté serveur pour Node.js

Selon [Asterisk 13 Events REST API](#), on sait qu'un appel de l'api REST Event est basé sur une WebSocket connection, donc on doit établir cette connection en utilisant WebSocket implémentation.

Pour l'application Web en temps réel, on utilise Socket.io.

6.2 Schéma



On s'installe Node.js dans une autre machine, elle a 2 rôles dans ce TP:

- être un WebSocket client (ws_client), elle connecte le serveur asterisk à distance via WebSocket connection et récupère des informations synchrone du serveur Asterisk.
- être un serveur HTTP qui attache une instance de socket.io pour utiliser la bibliothèque de socket.io pour la communication synchrone avec ses clients.

6.3 Mettre en place

- 1) installer Node.js dans une autre machine(windows 10 dans notre cas)
- 2) installer les packages WebSocket et Socket.io dans cette machine:

npm install websocket socket.io

- 3) créer le fichier app.js

```
var http = require('http');
var fs = require('fs');

// Chargement du fichier index.html affiché sur le client (navigateur)
var httpserver = http.createServer(function(req, res) {
  fs.readFile('./index.html', 'utf-8', function(error, content) {
    res.writeHead(200, {"Content-Type": "text/html"});
    res.end(content);
  });
});

// créer une connexion WebSocket au serveur Asterisk
const WebSocket = require('ws');
const ws_client = new
WebSocket('ws://192.168.141.235:8088/ari/events?api_key=toto:pass1&app=hello&
subscribeAll=true');
```

```

// ws_client écoute l'événement 'error' pour obtenir des erreurs
// lorsque le serveur Asterisk a des problèmes de connexion de socket
ws_client.on('error',function error(error){
    console.log(error);
})

// créer une instance socket.io attachée au serveur http
var io_httpserver = require('socket.io')(httpserver);

// écouter tous les clients qui se connectent au socket 'mysocket',
// dans ce cas, nous n'avons qu'un seul client (sokcet.io_client_1) dans
index.html
io_httpserver.sockets.on('connection', function (mysocket) {
    // ws_client écoute l'événement 'message' pour obtenir les données du
    serveur Asterisk
    ws_client.on('message',function show(data){
        // envoie des données à tous les clients qui écoutent l'événement
        personnalisé 'titi'
        mysocket.emit('titi',data);
    })
});

httpserver.listen(8080);

```

4) créer le fichier index.html

```

<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <title>Socket.io</title>
    <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.mi
n.css" />
  </head>

  <body>
    <h1>Communication avec socket.io !</h1>
    <table id="one" class="table table-dark">
      <thead>
        <tr>
          <th scope="col">date</th>
          <th scope="col">endpoints</th>

```

```

        <th scope="col">states</th>
    </tr>
</thead>
<tbody>
</tbody>
</table>
<script
src="https://cdn.jsdelivr.net/npm/socket.io-client@2/dist/socket.io.js
"></script>
<script>
    var tab = document.getElementById('one');
    var sokcet_io_client_1 = io.connect('http://localhost:8080');
    sokcet_io_client_1.on('titi', function(data) {
        console.log(data)
        var obj = JSON.parse(data);
        if(obj.type=="DeviceStateChanged"){
            var date =
obj.timestamp.split('.')[0].replace("T", " ");
            tab.insertAdjacentHTML('beforeend',
'<td>'+date+'</td><td>'+obj.device_state.name+'</td><td>'+obj.device_s
tate.state+'</td>');
        }
    })

</script>
</body>
</html>

```

- 5) lancer app.js
node app.js

6.4 Résultat

Quand on lance **node app.js** dans la machine de Windows 10(@ip192.168.141.236), on peut voir l'application Stasis 'hello' (qui a été définie dans app.js) est bien générée et activée dans le serveur Asterisk (@ip192.168.141.235):

```

Connected to Asterisk 13.14.1~dfsg-2+deb9u4 currently running on
debian (pid = 4741)
Creating Stasis app 'hello'
== WebSocket connection from '192.168.141.236:52056' for
protocol '' accepted using version '13'
Activating Stasis app 'hello'

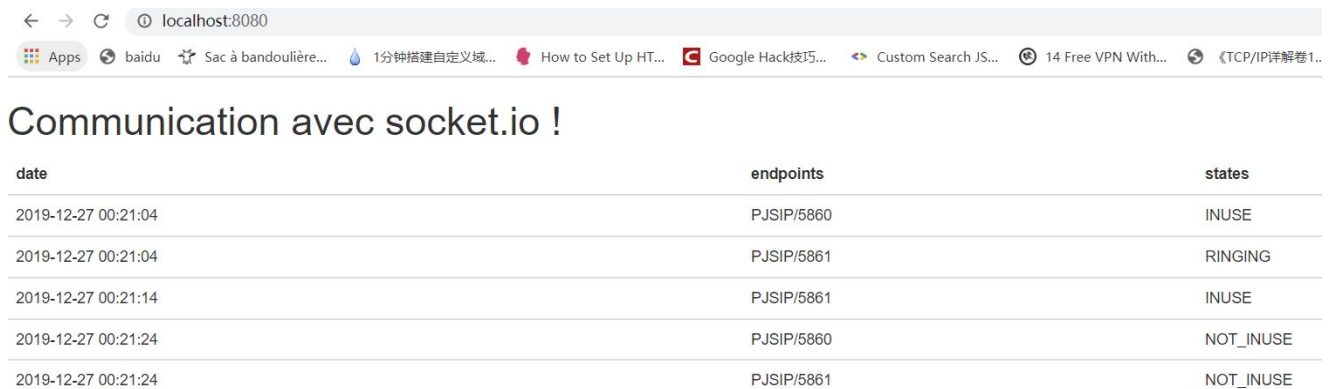
```

```

== Setting global variable 'SIPDOMAIN' to '192.168.141.235'
-- Executing [61@internes:1] Set("PJSIP/5860-00000000",
"VERIF=") in new stack
-- Executing [61@internes:2] GotoIf("PJSIP/5860-00000000",
"1?:renvoi") in new stack
-- Executing [61@internes:3] Dial("PJSIP/5860-00000000",
"PJSIP/5861,20") in new stack
-- Called PJSIP/5861
-- PJSIP/5861-00000001 is ringing

```

On ouvre localhost:8080/index.html dans un navigateur de la machine de Windows 10:



date	endpoints	states
2019-12-27 00:21:04	PJSIP/5860	INUSE
2019-12-27 00:21:04	PJSIP/5861	RINGING
2019-12-27 00:21:14	PJSIP/5861	INUSE
2019-12-27 00:21:24	PJSIP/5860	NOT_INUSE
2019-12-27 00:21:24	PJSIP/5861	NOT_INUSE

- 00:21:04: l'appelant 60 appelle 61, l'appelé 61 sonne.
- 00:21:14: l'appelé 61 s'occupe.
- 00:21:24 l'appelant et l'appelé sont disponibles.