Simphy: l'importanza della sincronizzazione nelle reti di comunicazione

(bozza preliminare)

Progetto n. 001 L'importanza della sincronizzazione nelle reti di comunicazione	Scheda n.01x
Progettista 5IA 2016-17 – Paolo Macchi -	*- 2016.12.11 requirements* *- 2017.06.09 rel.17.06.09*

Titolo
Simphy

Abstract
The importance of synchronization in the communication networks.

Playing a symphony in the cloud (and in the fog).
The project is a simple didactic application for the study of problems related to network synchronization and, in particular, is concerned with issues related to IoT.
The system involves the realization of an orchestra whose orchestras are automatic systems (Raspberry PI) synchronized by an "orchestra conductor" to play a symphony.
Client-Server architectures is the basis of the project and provides a protocol for sharing the score and start time from the server to the clients that request it.
Each client-orchestral will play a symphony track, related to the instrument that is right, at the precise time specified by the server-manager.

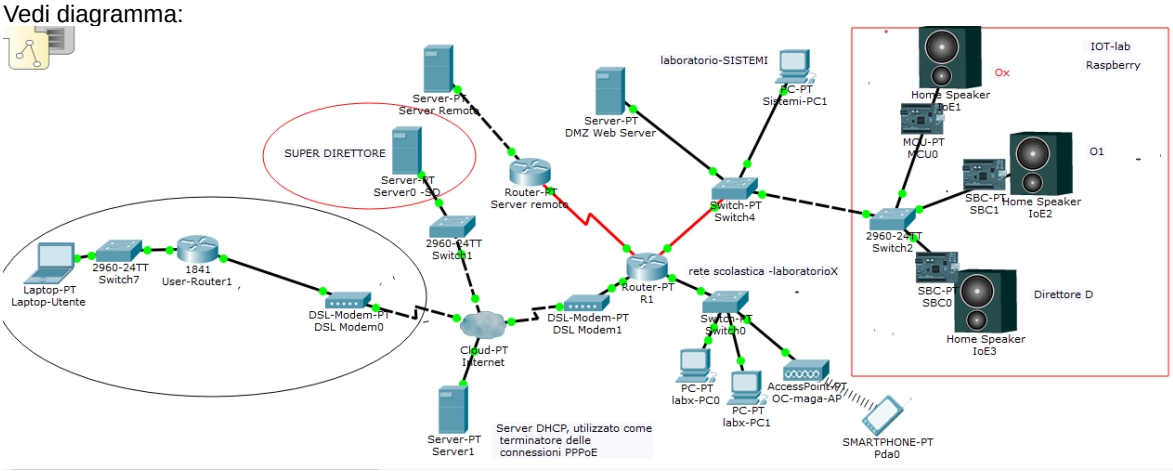
https://www.youtube.com/watch?v=HOt4-FzlHQs&feature=youtu.be

Obiettivi
Suonare una sinfonia in cloud (e in fog)

Descrizione
Il progetto è una semplice applicazione didattica per lo studio dei problemi legati alla sincronizzazione in rete e, in particolare, si interessa delle problematiche legate a IoT.

Il sistema prevede la realizzazione di un'orchestra i cui orchestrali sono sistemi automatici (Raspberry PI) sincronizzati da un "direttore d'orchestra", per suonare una sinfonia .
L'architetture Client-Server sta alla base del progetto e prevede un protocollo per lo scambio degli spartiti e dell'ora di inizio da parte del server ai client che ne fanno richiesta.
Ogni client-orchestrale suonerà una traccia della sinfonia, relativa allo strumento che gli è proprio, nel momento preciso indicato dal direttore-server.

Vedi diagramma:



Fasi del progetto e tempistica (allegare diagramma)
FASE 1

1. il "SuperDirettore" (SD) in Cloud decide la sinfonia da suonare e invia al Direttore (in fog) lo spartito e l'orario di inizio
2. il Direttore (D) distribuisce agli orchestrali (Ox) gli spartiti specifici relativi allo strumento da suonare (ad esempio O1-violino, O2-tromba, O3-timpani,..)
3. il Direttore, all'ora prestabilita, dà il via all'esecuzione

4. Ox suonano

FASE 2

1. l “SuperDirettore” (SD) in Cloud decide la sinfonia da suonare e invia ai Direttori (in fog) lo spartito e l’orario di inizio
2. come sopra ma con la sincronizzazione su server NTP (Network Time Protocol un protocollo per sincronizzare l'ora attraverso la rete: un client richiede l'ora corrente a un server e usa questa per impostare il proprio orologio.
Ad esempio http://www.pool.ntp.org/zone/it https://help.ubuntu.com/lts/serverguide/NTP.html net time /setsntp:ntp.ien.it).

Risorse e strumenti
Raspberry PI, Python, Fogging network, Cloud

Risultati attesi in termini di prodotto/servizi e rapporto costo/benefici

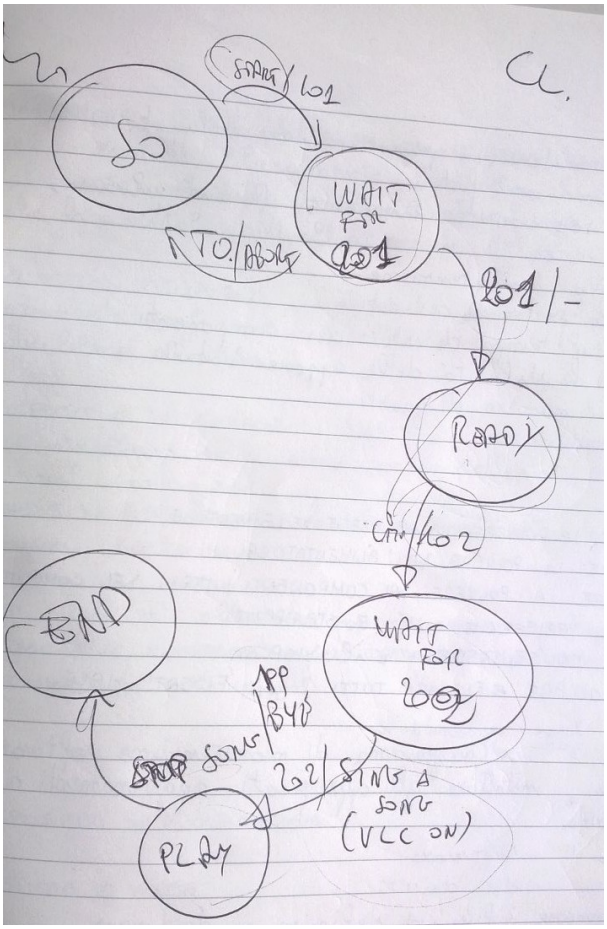
1. lavoro di squadra
2. progettazione e realizzazione software
3. sperimentare il cloud, il fogging, le reti

Note di progetto
Il progetto utilizza un modello a spirale con la realizzazione di prototipi successivi e con il relativo test

Note implementative

Codice client	Codice server	Azione	Comando da inserire client
101	201	Hello	
102	202	Prepara canzone	p=vlc.MediaPlayer("nomecanzone.mp3")
103	203	Parti canzone	p.play()
104	204	Pausa/Riprendi	p.pause()
105	205	Stop	p.stop()
199	299	Chiusura	

CLIENT

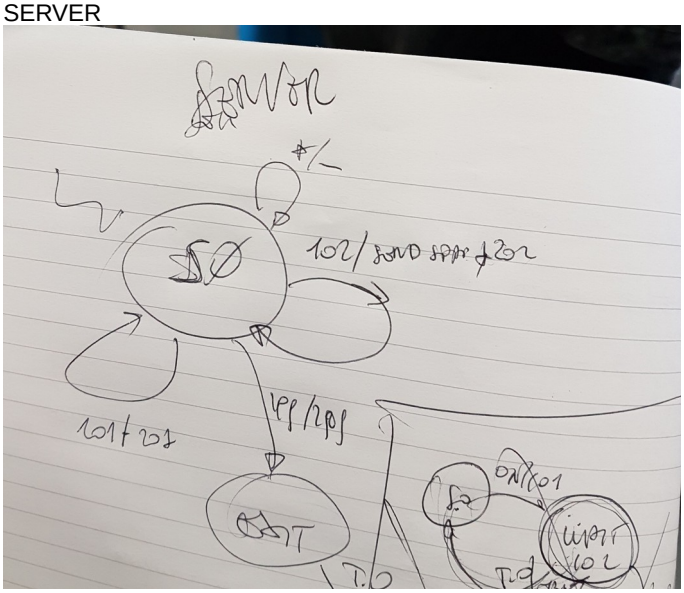



```
#!usr/bin/python3
import vlc
import socket
import RPi.GPIO as GPIO
from datetime import datetime,date,time,timedelta
##GPIO.setmode(GPIO.BCM)
##GPIO.setup(4,GPIO.OUT) ##led
##GPIO.output(4,0)
print "[CLIENT] Lettura configurazione"
file=open("conf","r")
testo = file.read()
file.close()
aTesto=testo.split( )
for a in aTesto:
        confSplit=a.split('=')
        if(confSplit[0]=="server"):
                server=confSplit[1]
        elif(confSplit[0]=="port"):
                port=int(confSplit[1])
print "[CLIENT] Connessione al server "+server+":"+str(port)+" in corso"
try:
        cs=socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        cs.connect((server,port))
except socket.error, ex:
        print "[CLIENT] ERRORE - eccezione del socket: "+str(ex)
        quit()

print "[CLIENT] Connesso al server"
##GPIO.output(4,1)
msgC=raw_input()
cs.sendto(msgC.encode(),(server,port))
msgS=cs.recv(1024)
print "[CLIENT] Ricevuto messaggio: "+msgS
if(msgS=="201"):
    msgC=raw_input()
```

```
	cs.sendto(msgC.encode(),(server,port))
	msgS2=cs.recv(1024)
	print "[CLIENT] Ricevuto messaggio: "+msgS2
if(msgS2=="202"):
	print "[CLIENT] preparo song1"
	p=vlc.MediaPlayer("song1.mp3")
	print "[CLIENT] Vuoi far partire la song? 103=SI"
	msgC=0
while(msgC!="199"):
	##print "Sono dentro"
	msgC=raw_input()
	cs.sendto(msgC.encode(),(server,port))
	msgS3=cs.recv(1024)
	print "[CLIENT] Ricevuto messaggio: "+msgS3
	if(msgS3.split('/')[0]=="203"):
		cs.close();
		d=datetime.strptime(msgS3.split('/')[1],'%Y-%m-%d %H:%M:%S');
		print d;
		while (d-datetime.now()).total_seconds()>0:
			print datetime.now()
			print (d-datetime.now())
		p.play()
		print "play"
		cs=socket.socket(socket.AF_INET, socket.SOCK_STREAM)
		cs.connect((server,port))
	if(msgS3=="204"):
		p.pause()
		print "Pausa"
	if(msgS3=="205"):
		p.stop()
		print "Stop"
```

SERVER



```
import socket
from datetime import datetime,date,time
print "[SERVER] Avvio del server in corso"
ss=socket.socket(socket.AF_INET, socket.SOCK_STREAM)
print "Il tuo IP: "+socket.gethostbyname(socket.gethostname())
ss.bind(("192.168.101.13",15150))
ss.listen(5)
while True:
		print "[SERVER] In ascolto"
		try:
			cs, serverAddress=ss.accept()
			print "[SERVER] Nuova connessione da "+cs.getpeername()[0]+":"+str(cs.getpeername()[1])
			serverLogString="[SERVER-"+cs.getpeername()[0]+":"+str(cs.getpeername()[1])+"] "
			chiusura=True
			while chiusura:
```

```
                print serverLogString+"In attesa di messaggio"
                dati=cs.recv(1024)
                print serverLogString+"Ricevuto messaggio: "+dati
                msg="404"
                if(dati=="101"):
                        msg="201"
                elif(dati=="199"):
                        msg="299"
                elif(dati=="102"):
                        msg="202"
                elif(dati=="103"):
                        msg="203/"+str(datetime(2017,06,7,13,39,00));
                elif(dati=="104"):
                        msg="204"
                elif(dati=="105"):
                        msg="205"
                print serverLogString+"Invio messaggio: "+msg
                cs.sendto(msg,(cs.getpeername()[0], cs.getpeername()[1]))
                if(dati=="199"):
                        print serverLogString+"Chiusura della connessione"
                        chiusura=False
                        cs.close()
        except socket.error, ex:
                print "[CLIENT] ERRORE - eccezione del socket: "+str(ex)
```