```python
#!/usr/bin/env python3
import asyncio
import websockets
import json

# initialisation du tableau associatif pour le premier envoi
messageEnvoye = {
    'texte': 'Bonjour ',
    'numero': 1,
  };

# Gestion de l'envoi des messages du serveur au client
async def gestion_envoi_message(websocket):
    # boucle infinie pour envoyer toutes les secondes un message ;
    while True:
       # formatage du dictionnaire au format JSON
       messageEnvoyeJSON = json.dumps(messageEnvoye)
       # envoi du message au format JSON
       await websocket.send(messageEnvoyeJSON)
       # incrémenter le numéro du message
       messageEnvoye["numero"] = messageEnvoye["numero"] + 1
       # attendre une seconde
       await asyncio.sleep(1)

# Gestion des messages recus du client au format JSON
async def gestion_reception_message(websocket):
    while True:
        # reception des messages d'un client
        messageRecuJSON = await websocket.recv()
        # conversion du message format JSON en dictionnaire Python
        messageRecu=json.loads(messageRecuJSON)
        print(" {} {} {}".format(messageRecu["texteDebut"],messageRecu["numero"],messageRecu["texteFin"],))

# fonction lancee à chaque connexion d'un client
async def echange(websocket,path): # définir la fonction comme asynchrone
    #envoyer des messages en parallèle
    envoyer = asyncio.ensure_future(gestion_envoi_message(websocket))
    #recevoir message en parallèle
    recevoir = asyncio.ensure_future(gestion_reception_message(websocket))
    termine, attente = await asyncio.wait(
            [envoyer, recevoir],
            return_when = asyncio.FIRST_COMPLETED,
    )

# Definir la fonction qui sera appelee par le serveur a la connexion d'un client
lancement_serveur = websockets.serve(echange,'10.3.141.1', 5678)
# Creation de la boucle d'evenement (event loop)
loop = asyncio.get_event_loop()
loop.run_until_complete(lancement_serveur)
loop.run_forever()
loop.close()
```