```
def mpc_cal(Bandwidth, total, rate, buffer, remain, Send, Size, index, M, final_rate):
    if total==tau:
        for i in range(tau): #choose the tau-th bitrate
            priority = False
            Frame = rate[i]//fps
            Num = Num + rate[i]/10000.0/tau
            for j in range(GoP):
                if priority: ### drop unencodable frames
                    drop = drop + 1
                else:
                if remain<Bandwidth[j+i*GoP]:
                    rest = Bandwidth[j+i*GoP] - remain ### send frames as many as possible
                    while len(Send) and rest>0:
                    remain = 0
                else:
                    remain = remain - Bandwidth[j+i*GoP]
                buffer = max(buffer-Bandwidth[j+i*GoP],0)
                if len(Send) and GoP*(index+i)+j- min(Send)+1>=Bmax: ## voliate the timeliness rule, drop strategy
                count=0
                if priority: ## drop unencodable frames
                    for block in range(len(Send)):
                Send = Send[count:]
                Size = Size[count:]
        Num = Num - beta*drop
        return (final_rate,M)
    else:
        for i in range(len(Bitrate)): ### search all the possible choice tau times
            final_rate,M = mpc_cal(Bandwidth,total+1,rate,buffer,remain,New_Send,New_Size,index,M,final_rate)
        return (final_rate,M)
```