```
/*
 * CsmaMac.h
 *
 */

#ifndef CSMAMAC_H_
#define CSMAMAC_H_

#include <omnetpp.h>
#include "CSResponse_m.h"
#include "TransmissionConfirmation_m.h"
#include "TransmissionIndication_m.h"
#include "AppMessage_m.h"
#include "MacPacket_m.h"
// further includes ...


using namespace omnetpp;


// design a state machine and suitable states, name them here ..
enum MacState {
  IDLE = 0,
  TRANSMIT = 1,
  SENSE = 2,
  CSBACKOFF = 3,
  RETRANSMIT = 4,
  ACK_LISTEN = 5
};


// string names for your states (for debugging purposes)
const char * const stateStrings[] = {
  "Idling", "Transmitting", "Carrier sensing", "Backing off", "Retransmission",
"Listening for ack"
};


class CsmaMac : public cSimpleModule {

public:
  // your public methods and data members
    void initialize();
    void handleMessage(cMessage* msg);
    ~CsmaMac();

protected:

  // your protected methods and data members
    int ownAddress;
int bufferSize;
int maxBackoffs;
int maxAttempts;
int macOverheadSizeData;
int macOverheadSizeAck;   // Overhead for a MAC acknowledgement
double macAckDelay;             // fixed waiting time for sending ACK
double ackTimeout;

// gates
int gidFromApp;
int gidToApp;
int gidFromXcvr;
int gidToXcvr;

// self messages
MacPacket* currMsg;
MacPacket* currAckMsg;
cMessage* csBackoffCompMsg;
cMessage* retransmitCompMsg;
cMessage* ackTimeoutMsg;
cMessage* macAckDelayExpired;
cMessage* succAttBackoffMsg;

// buffer
cQueue macBuff;

MacState state;
int attemptCnt;
int backoffCnt;
bool ackSent;

volatile double csBackoff;
volatile double attBackoff;
volatile double succBackoff;

void handleCSResponse(CSResponse* csResp);
void handleTransmissionConfirmation(TransmissionConfirmation* transConf);
void handleTransmissionIndication(TransmissionIndication* transInd);
void handleAppMessage(AppMessage* appMsg);
void beginDataMAC(void);
void attemptTransmit(void);
virtual void carrierSenseAttempt(void);
void handleDataPkt(MacPacket* decapMsg);
void handleAckPkt(MacPacket* decapMsg);
void handleAckTimeout(void);
void handleRetrasmitComp(void);
void handleFailedTransmission(void);
void handleCsBackoff(void);
void handleMaxBackoff(void);
void sendAckMAC(void);
```

```
    void handleSuccBackoff(void);
};


#endif /* CSMAMAC_H_ */
```