**Responsibilities**

*Axis API*: The distributed API, responsible for actually capturing images.

*FrameGrabberThread*: thread responsible for processing images and detecting motion.

*CameraControl (Monitor) & OutputThread(s)/Input-thread*: monitor class, containing methods used in the thread above. Also contains methods output thread(s)/input thread, i.e. thread(s) that send the received images/handles commands from the model.

*ReaderThread(s)*: Thread(s) responsible for reading the output from the output thread(s).

*Model (Monitor)*: monitor class, containing methods used by the ReaderThread. The Model-class is also the model on which the GUI (below) depends, as in a model-view architecture.

*ViewingInstance, Image processing thread & GUI*: ViewingInstance is an object (instance) created out of the image received, handled by the image processing thread class and later put in the GUI. The GUI also sends commands put in by the user to the model.

*SignalingThread*: Handles commands by the user and sends them to the InputThread on the ”Server-side”.

*ProtocolTranslator(TCP)*: a parser class, parsing the data as needed.

**Moviemode propagation**

Our idea is that once a camera detects motion, it should be able to send a message via the CameraControl to the respective OutputThread, which is then received by the respective ReaderThread using a method in the Model-class. This method signals the SignalingThread, which in turn signals all the cameras that moviemode is to be turned on.

Network protocol

Byte 1 – Type

***TYPE:{ A(1) , B(2) , C(4) }***  
A(1): picture  
Byte 2 – Timestamp  
Byte 3 - Size  
Byte rest – Image  
  
B(2): movie mode message

C(4): idle mode message

TIMESTAMP:  
Milliseconds since the last whole numbers of minutes since the system time started in 1970.  
A.K.A. (SystemTime)%(60\*1000)= timestamp.

SIZE:  
Depending on type size = image size. If movie/idle mode size is not needed.

