University of York

Embedded Systems Design & Implementation

Open Individual Assessment

Open Assessment 1

Examination number:

Y3606797

Contents

1 Part 1 - Theory

1.1 Question 1

We can determine the rate X of actor H by producing a set of simultaneous equations from Table 1 and the provided Synchronous Dataflow model.

The topology matrix for the SDF model is as follows:

$$\Gamma = \begin{bmatrix} 2 & 0 & 0 & 0 & -2 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & -2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & -2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & -2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & -6 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 3 & -2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & X & -3 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 \end{bmatrix}$$

This gives us the following simultaneous equations:

$$\begin{array}{lll} 2A - 2E = 0 & 2B - 2E = 0 & 2C - 2E = 0 \\ 2D - 2E = 0 & 2E - 6F = 0 & F - I = 0 \\ 3G - 2H = 0 & XH - 3I = 0 & I - G = 0 \end{array}$$

Using these equations I determined that $X = 2$. Similarly, I determined the firing frequencies of the remaining actors, seen in the vector q:

$$q = \begin{matrix} A \\ B \\ C \\ D \\ E \\ F \\ G \\ H \\ I \end{matrix} \begin{pmatrix} 6 \\ 6 \\ 6 \\ 6 \\ 6 \\ 2 \\ 2 \\ 3 \\ 2 \end{pmatrix}$$

1.2 Question 2

Using the firing frequencies determined in Question 1, I was able to identify the following PASS schedule:

a.fire(3); b.fire(3); c.fire(3); d.fire(3); e.fire(3); f.fire(1); a.fire(3); b.fire(3); c.fire(3); d.fire(3); e.fire(3); f.fire(1); g.fire(2); h.fire(3); i.fire(2);

The maximum required FIFO buffer size is 6 as required and the number of firings of the actors match up with their frequencies in the vector q (Question 1).

1.3 Question 3

For my chosen PASS schedule the number of tokens that must be initially stored in the buffer of the feedback channel c9 is 2.

2 Part 2 - WSN MAC layer protocol

2.1 Question 1

For this task I opted to dedicate an entire class (SourceNodeActor) to perform the PtolemyII actor functions needed in the simulation. The class SourceNodeActor relies on another class (SourceNode) which provides the implementation of the protocol features, this in turn relies on another class SinkNodeModel which is responsible for modelling the sink nodes parameters and ultimately synchronising with their reception phases.

An instance of SourceNode is created with a number of channels provided to the constructor (SourceNodeActor, initialise, 62). This design was taken so that I could reuse the code during the second

part of this question.

When the actor is initialised it begins reading beacons from a sink node channel for a specified amount of time, given by the constant DEFAULT_LISTENING_TIME (SourceNode, 11), when we read a beacon we extend the time to try get another and thus calculate the sink parameters. If the time expires we switch nodes and do the same again - this is how I sync to multiple nodes effectively.

When n=1 we have to wait an entire protocol length to read another beacon, so we can switch away and come back later (SourceNode, readBeacon, 64). We then use the length between the iterations to calculate the sink parameters (SinkNodeModel, calcTForNEqual1, 115-119) and (SinkNodeModel, calcN, 102-109).

2.2 Question 2

3 Part 3 - Embedded platform modelling

3.1 Question 1