Automatic Stock Notification

Sensor Network Lab 14/15 - Group 3
Abhishek Srujan, Arun Prabhu, Ingo Larch, Tamilselvan Shanmugam

Overview

- Sensor nodes are attached to products
- Whenever products are taken out for delivery the sensor detects the change
- Furthermore, the temperature of products is monitored
- Nodes forward information (i.e. temp. value) to nearby router
- Stock changes and temperature values are stored centrally in a DB
- A user-interface allows to access real-time information

Goal and benefits

- System to keep track of current stock in warehouse
- Real-time information is available via an interface so that staff or other users along the supply-chain can monitor products
- Helps in preventing wastage due to overproduction and meet the market demand correctly
- The temperature monitoring guarantees that products are stored at ideal conditions with the aim to increase shelf time

Hardware and Software

Sensor Nodes

Tmote Sky, TelosB running TinyOS

(with build-in temperature sensor, LEDs)

Router

Tmote Sky, TelosB running TinyOS

Storage

MySQL Database

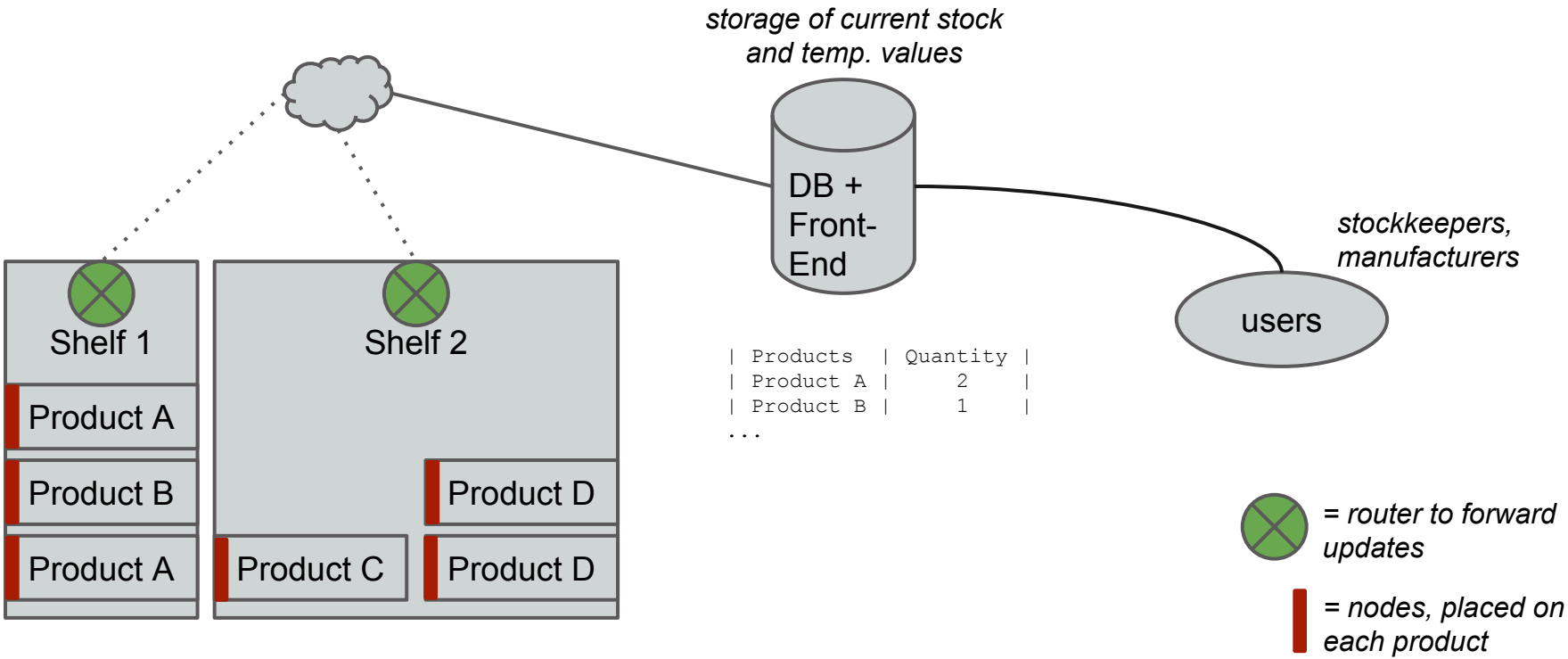
User Interface

HTML/PHP

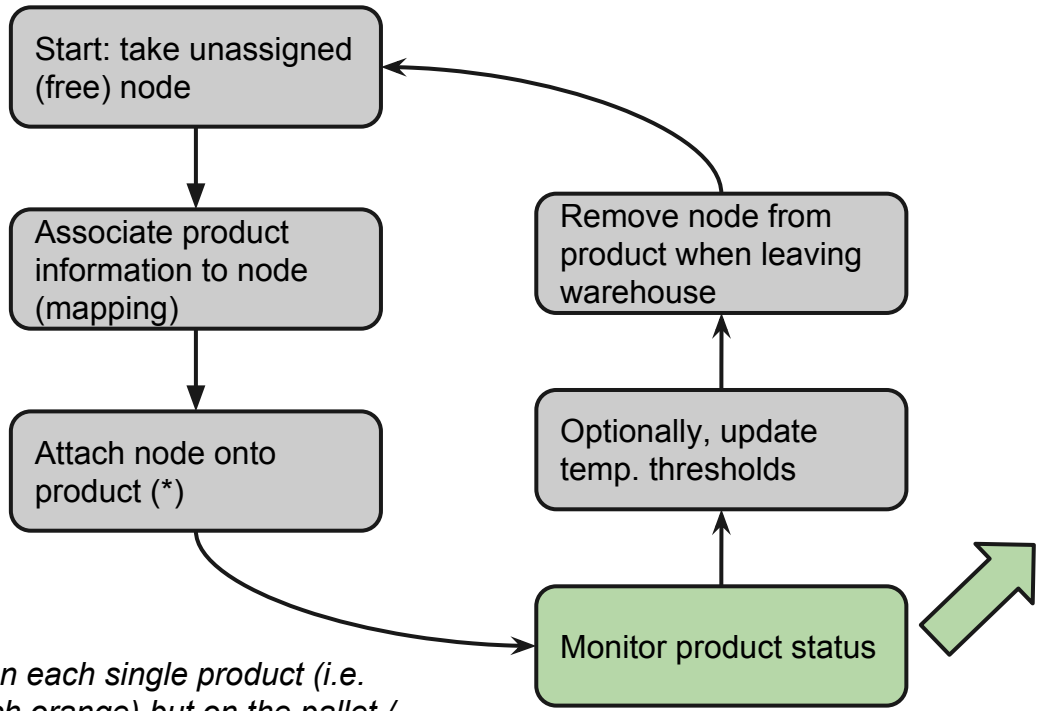
Features

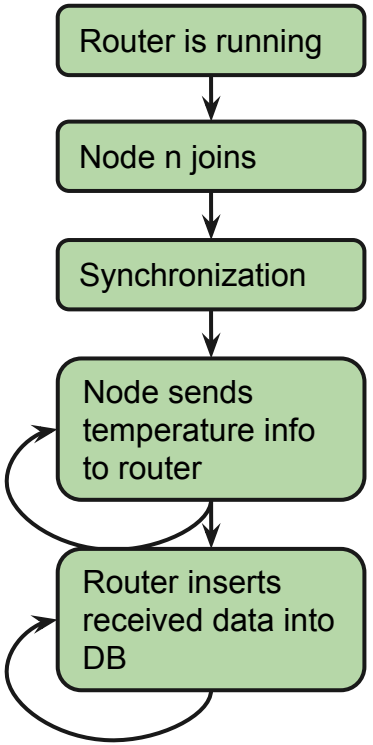
Sensing	→	**Reading Temperature values** *Make sure product is stored at right/best temperature*
Computation	→	**Presence of a Product** *Allows to monitor if a particular product is present or if the temp. is above a threshold*
Communication	→	**A Node sends info to router** *Allows to send information and store it centrally into a DB*

Architecture



Workflow

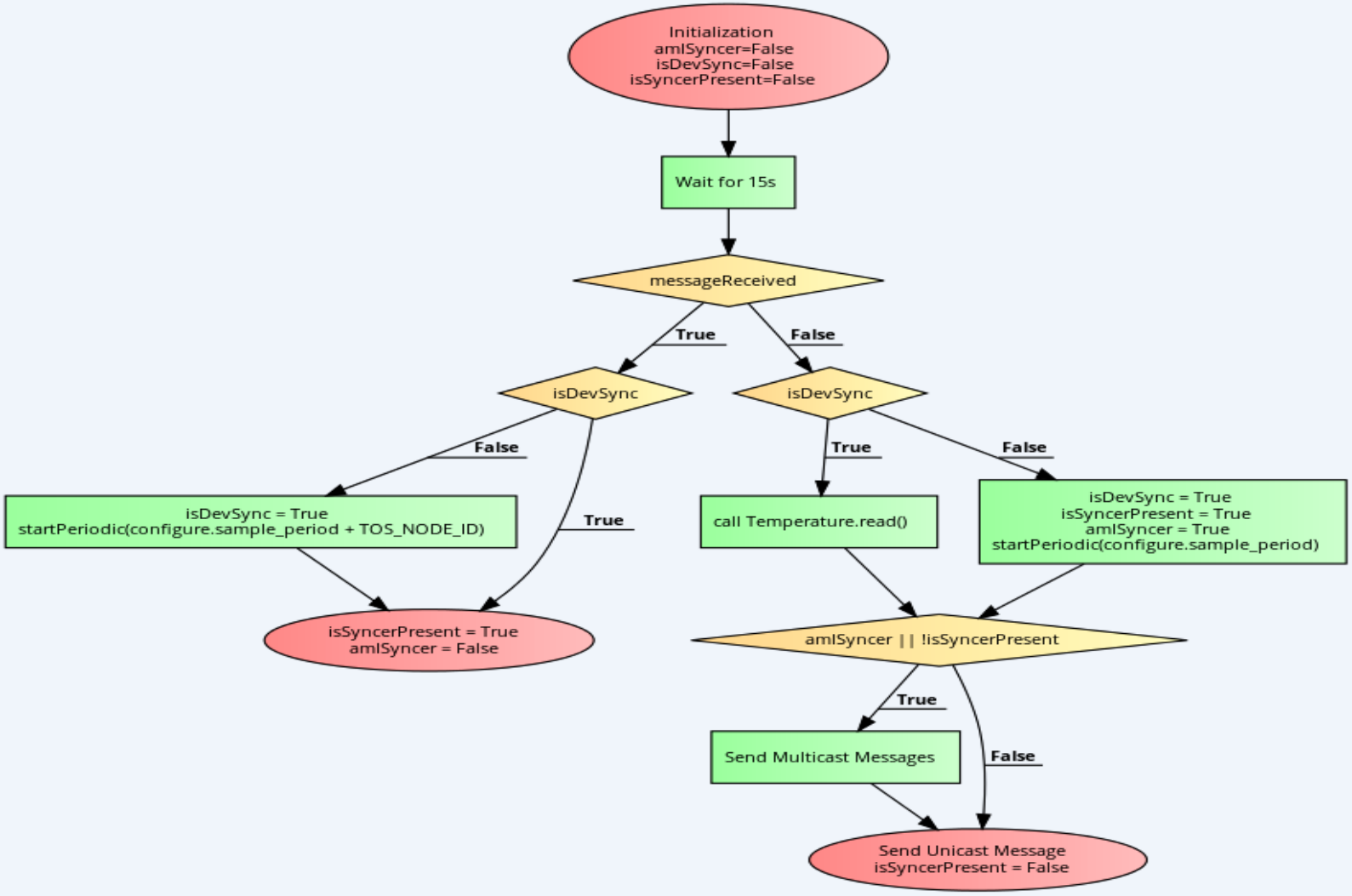




** not on each single product (i.e. on each orange) but on the pallet / container / box they are contained*

Implementation

Synchronization


Initialization
amISyncer=False
isDevSync=False
isSyncerPresent=False
Wait for 15s
messageReceived
True
False
isDevSync
isDevSync
False
True
False
isDevSync = True
startPeriodic(configure.sample_period + TOS_NODE_ID)
True
call Temperature.read()
isDevSync = True
isSyncerPresent = True
amISyncer = True
startPeriodic(configure.sample_period)
isSyncerPresent = True
amISyncer = False
amISyncer || !isSyncerPresent
True
False
Send Multicast Messages
Send Unicast Message
isSyncerPresent = False

Implementation

Inserting sensed values to DB

```
rpt = Sensing.Sensing(data=data, data_length=len(data))
nodeId = rpt.get_node_id()
productId = rpt.get_product_id()
temperature = rpt.get_temp()
warning = rpt.get_warning()
print 'Format: {nodeId}, {productId}, {temperature}, {warning}'.format(nodeId=nodeId, productId=productId,
temperature=temperature, warning=warning)
cursor.execute ('insert into product values("%s", "%s", "%s", "%s") ON DUPLICATE KEY UPDATE temperature = "%s", warning = "%s"'
% \(nodeId, productId, temperature, warning, temperature, warning))
connection.commit ()
```

Challenges

- Synchronisation
- Database Integration

Organisation of Work

Nodes

- associate to router
- sense temperature
- communicate temperature to router
- check if sensed temperature is above threshold
- show visually using the LED if temperature is too high
- Synchronization

Arun Prabhu, Tamilselvan Shanmugam

DB and Web Front-End

- DB schema that holds “static” data (products) and current available nodes
- create views for listing total amount of products

Router

- Maintain list of associated nodes
- Forward messages to DB
- Allows update of threshold values
- Detection of absence of a node

Abhishek Srujan, Ingo Larch

Other Tasks (design, integration,testing)
Arun Prabhu, Tamilselvan Shanmugam, Abhishek Srujan, Ingo Larch

Outlook

Possible future extension / improvements

- Keep historical information in DB for later analysis
- Add localization support to know exact place where a product is located
- More effective synchronization and collision avoidance scheme can be utilized
- Expand the application over several companies and sites, so that the product can be traced from a given manufacturer until reaching it’s final destination (i.e. supermarket)

Thanks for your attention