Lecture 8 – RTS Examples (Sommerville Ch. 13)

Karl R. Wilcox
K.R.Wilcox@reading.ac.uk

Objectives

- To introduce examples of real time system architectures for monitoring and control and data acquisition systems

Monitoring and control systems

- Important class of real-time systems
- Continuously check sensors and take actions depending on sensor values
- Monitoring systems examine sensors and report their results
- Control systems take sensor values and control hardware actuators

Burglar alarm system

- A system is required to monitor sensors on doors and windows to detect the presence of intruders in a building
- When a sensor indicates a break-in, the system switches on lights around the area and calls police automatically
- The system should include provision for operation without a mains power supply

Burglar alarm system

- **Sensors**
  - Movement detectors, window sensors, door sensors.
  - 50 window sensors, 30 door sensors and 200 movement detectors
  - Voltage drop sensor
- **Actions**
  - When an intruder is detected, police are called automatically.
  - Lights are switched on in rooms with active sensors.
  - An audible alarm is switched on.
  - The system switches automatically to backup power when a voltage drop is detected.

The R-T system design process

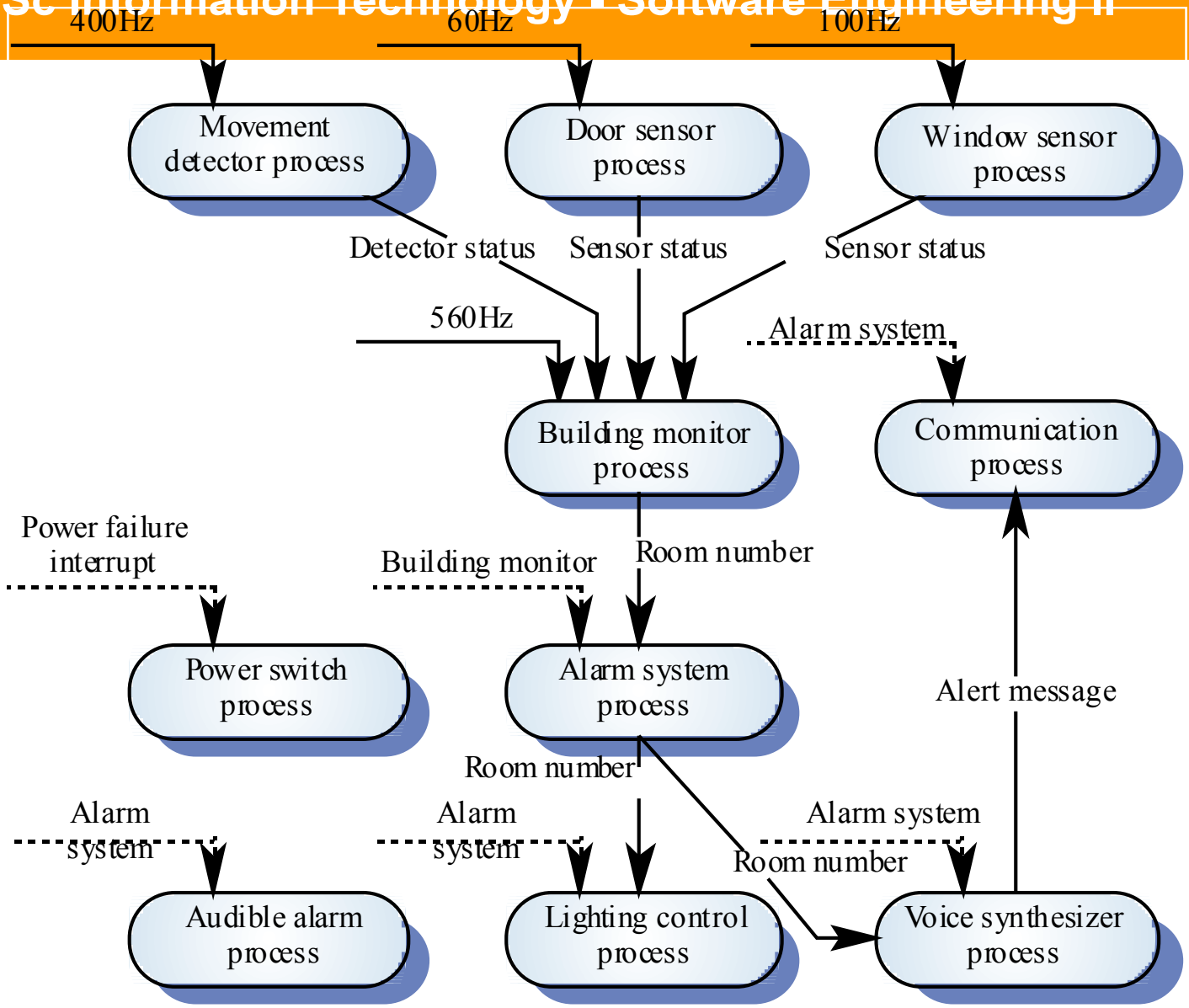
- Identify stimuli and associated responses
- Define the timing constraints associated with each stimulus and response
- Allocate system functions to concurrent processes
- Design algorithms for stimulus processing and response generation
- Design a scheduling system which ensures that processes will always be scheduled to meet their deadlines

Stimuli to be processed

- **Power failure**
  - **Generated aperiodically by a circuit monitor. When received, the system must switch to backup power within 50 ms**
- **Intruder alarm**
  - **Stimulus generated by system sensors. Response is to call the police, switch on building lights and the audible alarm**

Timing requirements

Stimulus/Response	Timing requirements
Power fail interrupt	The switch to backup power must be completed within a deadline of 50 ms.
Door alarm	Each door alarm should be polled twice per second.
Window alarm	Each window alarm should be polled twice per second.
Movement detector	Each movement detector should be polled twice per second.
Audible alarm	The audible alarm should be switched on within 1/2 second of an alarm being raised by a sensor.
Lights switch	The lights should be switched on within 1/2 second of an alarm being raised by a sensor.
Communications	The call to the police should be started within 2 seconds of an alarm being raised by a sensor.
Voice synthesiser	A synthesised message should be available within 4 seconds of an alarm being raised by a sensor.


400Hz
60Hz
100Hz
Movement detector process
Door sensor process
Window sensor process
Detector status
Sensor status
Sensor status
560Hz
Alarm system
Building monitor process
Communication process
Power failure interrupt
Building monitor
Room number
Power switch process
Alarm system process
Alert message
Room number
Alarm system
Alarm system
Alarm system
Room number
Audible alarm process
Lighting control process
Voice synthesizer process

```
// See http://www.software-engin.com/ for links to the complete Java code for this
// example

    class BuildingMonitor extends Thread {

        BuildingSensor win, door, move ;

        Siren      siren = new Siren () ;
        Lights     lights = new Lights () ;
        Synthesizer synthesizer = new Synthesizer () ;
        DoorSensors doors = new DoorSensors (30) ;
        WindowSensors    windows = new WindowSensors (50) ;
        MovementSensors movements = new MovementSensors (200) ;
        PowerMonitor pm = new PowerMonitor () ;

        BuildingMonitor()
        {
            // initialise all the sensors and start the processes
            siren.start () ; lights.start () ;
            synthesizer.start () ; windows.start () ;
            doors.start () ; movements.start () ; pm.start () ;

      }
```

Building_monitor process 1

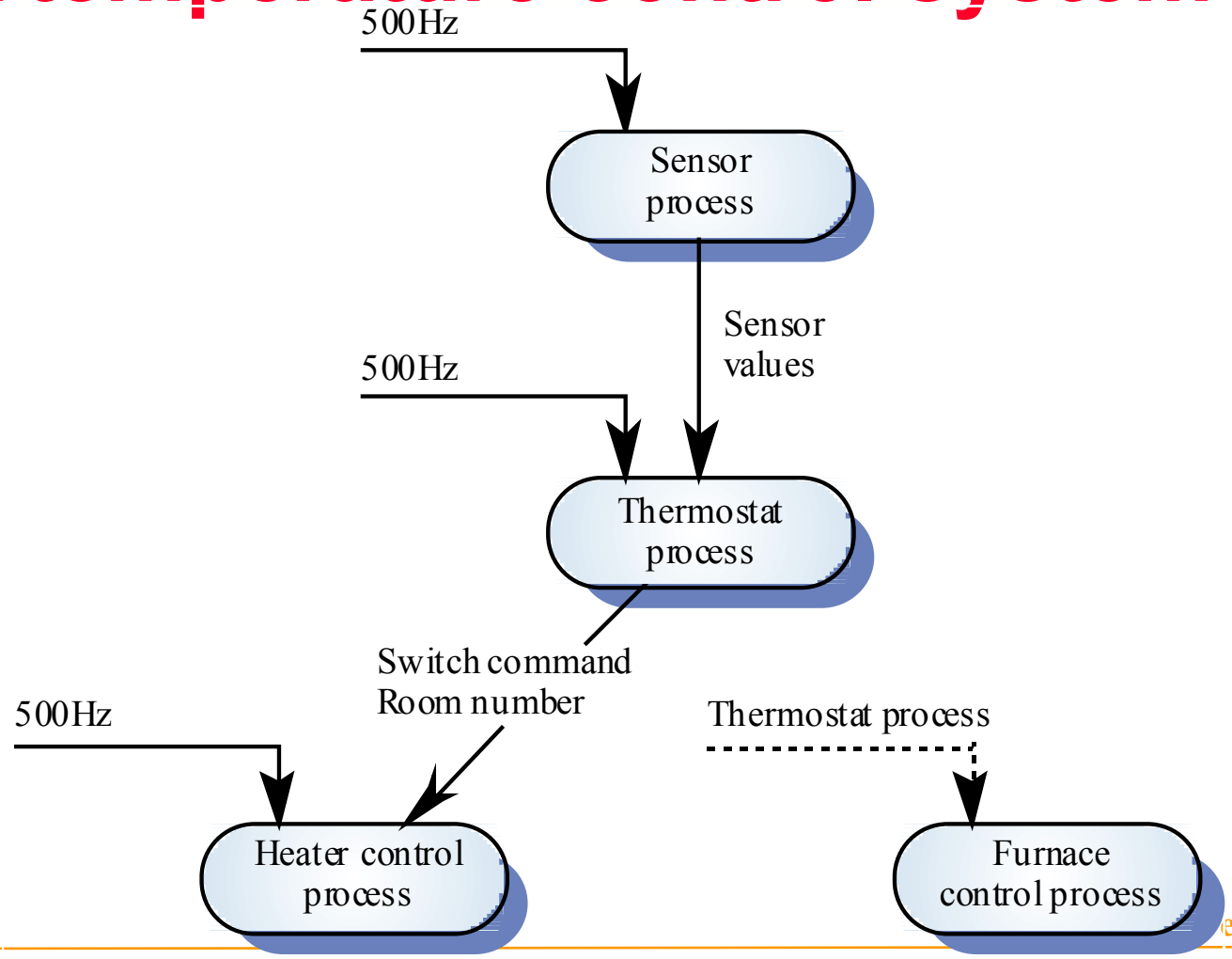
```
    public void run ()
    {
        int room = 0 ;
        while (true)
        {
            // poll the movement sensors at least twice per second (400 Hz)
            move = movements.getVal () ;
            // poll the window sensors at least twice/second (100 Hz)
            win = windows.getVal () ;
            // poll the door sensors at least twice per second (60 Hz)
            door = doors.getVal () ;
            if (move.sensorVal == 1 | door.sensorVal == 1 | win.sensorVal == 1)
                {
                    // a sensor has indicated an intruder
                    if (move.sensorVal == 1)      room = move.room ;
                    if (door.sensorVal == 1)      room = door.room ;
                    if (win.sensorVal == 1 )                  room = win.room ;

                    lights.on (room) ; siren.on () ; synthesizer.on (room) ;
                    break ;
                }
        }
        lights.shutdown () ; siren.shutdown () ; synthesizer.shutdown () ;
        windows.shutdown () ; doors.shutdown () ; movements.shutdown () ;

    } // run
} //BuildingMonitor
```

Building_monitor process 2

Control systems

- A burglar alarm system is primarily a monitoring system. It collects data from sensors but no real-time actuator control
- Control systems are similar but, in response to sensor values, the system sends control signals to actuators
- An example of a monitoring and control system is a system which monitors temperature and switches heaters on and off

A temperature control system



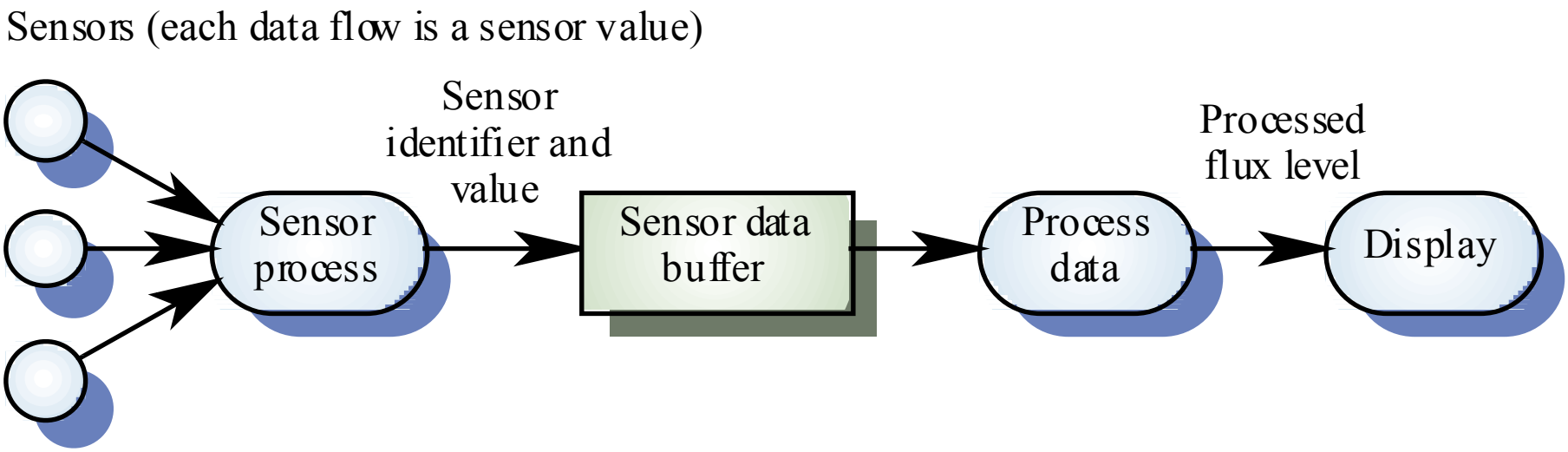
Data acquisition systems

- Collect data from sensors for subsequent processing and analysis.
- Data collection processes and processing processes may have different periods and deadlines.
- Data collection may be faster than processing e.g. collecting information about an explosion.
- Circular or ring buffers are a mechanism for smoothing speed differences.

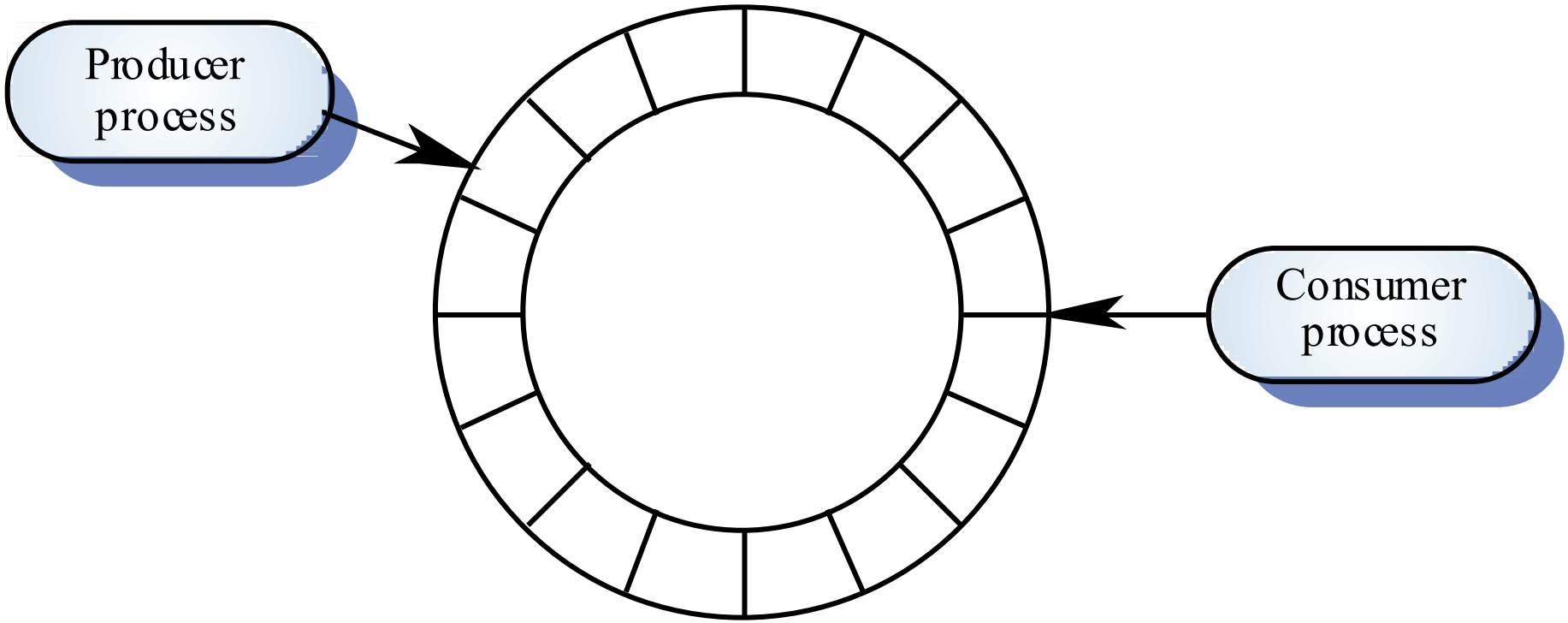
Reactor data collection

- A system collects data from a set of sensors monitoring the neutron flux from a nuclear reactor.
- Flux data is placed in a ring buffer for later processing.
- The ring buffer is itself implemented as a concurrent process so that the collection and processing processes may be synchronized.

Reactor flux monitoring



A ring buffer



Mutual exclusion

- Producer processes collect data and add it to the buffer. Consumer processes take data from the buffer and make elements available
- Producer and consumer processes must be mutually excluded from accessing the same element.
- The buffer must stop producer processes adding information to a full buffer and consumer processes trying to take information from an empty buffer.

```
class CircularBuffer
{
      int bufsize ;
      SensorRecord [] store ;
      int numberOfEntries = 0 ;
      int front = 0, back = 0 ;

      CircularBuffer (int n) {
            bufsize = n ;
            store = new SensorRecord [bufsize] ;
      } // CircularBuffer

      synchronized void put (SensorRecord rec ) throws InterruptedException
      {
            if ( numberOfEntries == bufsize)
                  wait () ;
            store [back] = new SensorRecord (rec.sensorId, rec.sensorVal) ;
            back = back + 1 ;
            if (back == bufsize)
                  back = 0 ;
            numberOfEntries = numberOfEntries + 1 ;
            notify () ;
      } // put
```


Java implementation of a ring buffer 1

```
 synchronized SensorRecord get () throws InterruptedException
{
        SensorRecord result = new SensorRecord (-1, -1) ;
        if (numberOfEntries == 0)
                wait () ;
        result = store [front] ;
        front = front + 1 ;
        if (front == bufsize)
              front = 0 ;
        numberOfEntries = numberOfEntries - 1 ;
        notify () ;
        return result ;
} // get
} // CircularBuffer
```