

# Awesomegotchi

Proximity Gaming for Social Networking

# Overview

- Mobile gaming increasingly popular
- Use of social networking increasing
- Location-aware applications

# Application

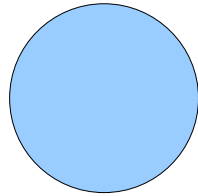
- Each Awesomegotchi device can interface with other Awesomegotchi devices to play games
- Base stations provide location and internet connectivity
- Possible applications in social studies of children (tracking social groups via who plays with who)
- Gaming application could be changed for other types of study

# Application

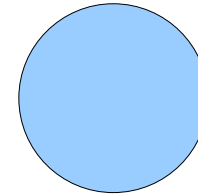
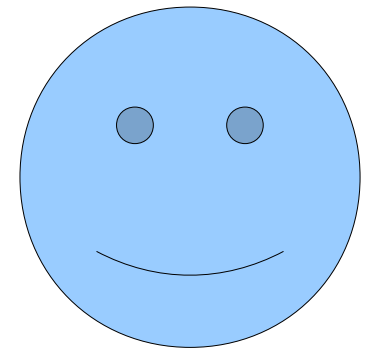
- Current implementation is basic but provides
  - Basic gaming
  - Mesh networking
  - Location information from base station

# Example Use

Base  
Station

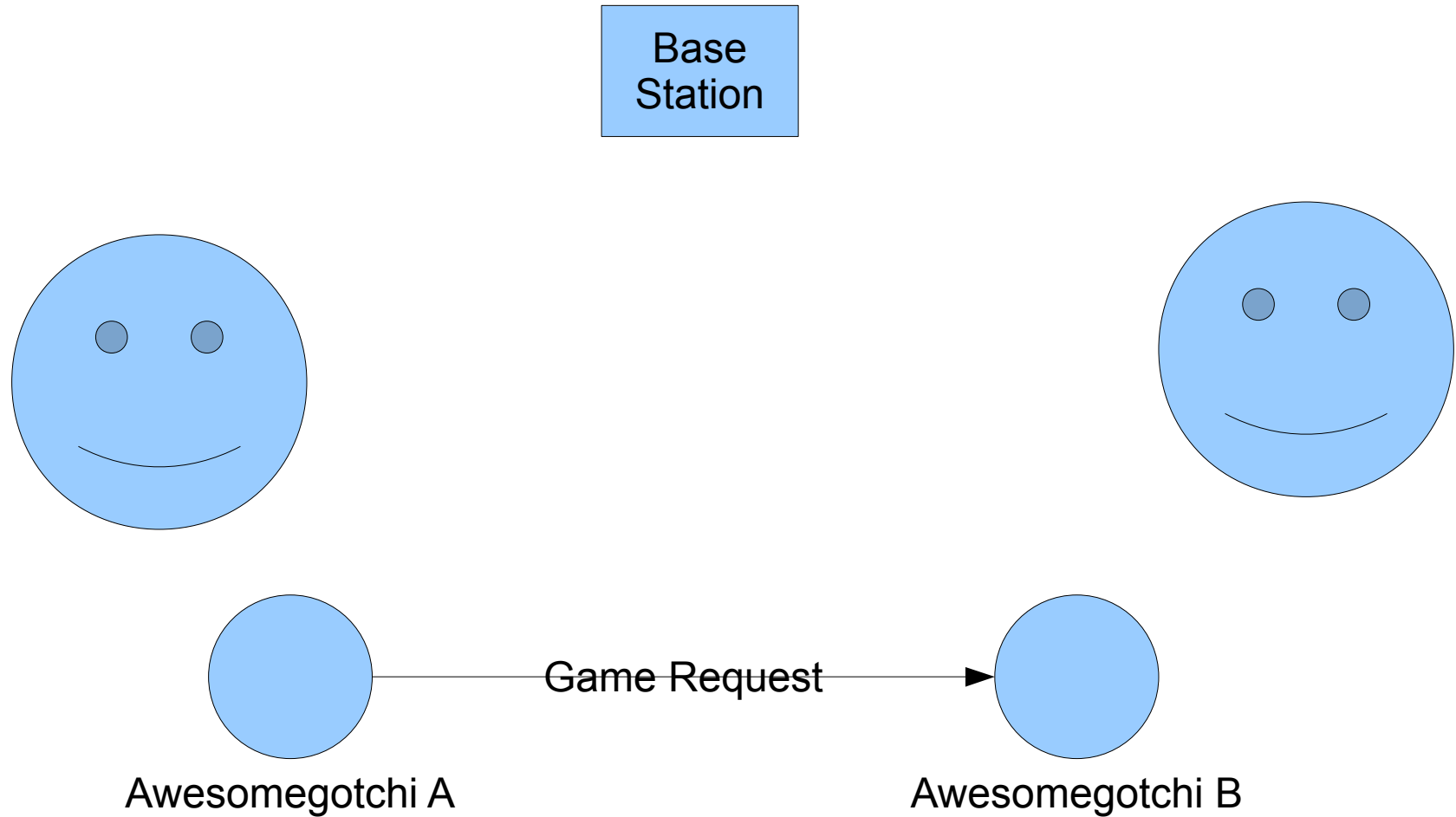


Awesomegotchi A



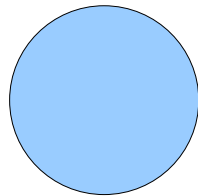
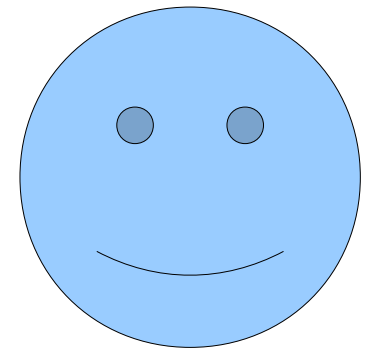
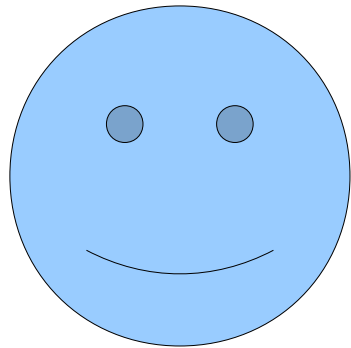
Awesomegotchi B

# Example Use

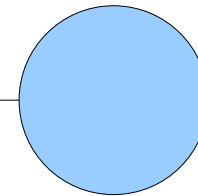


# Example Use

Base  
Station



Awesomegotchi A

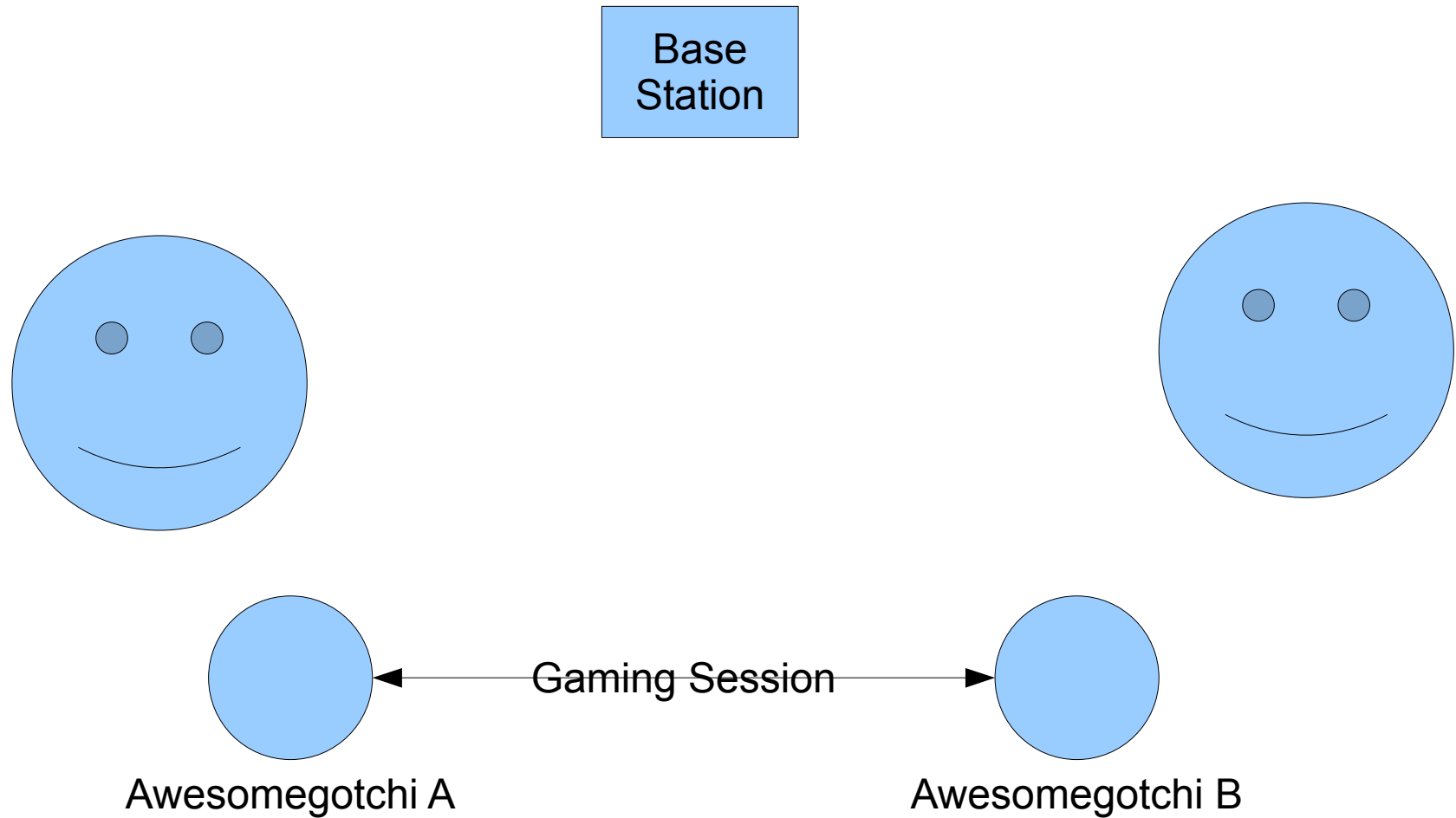


Awesomegotchi B

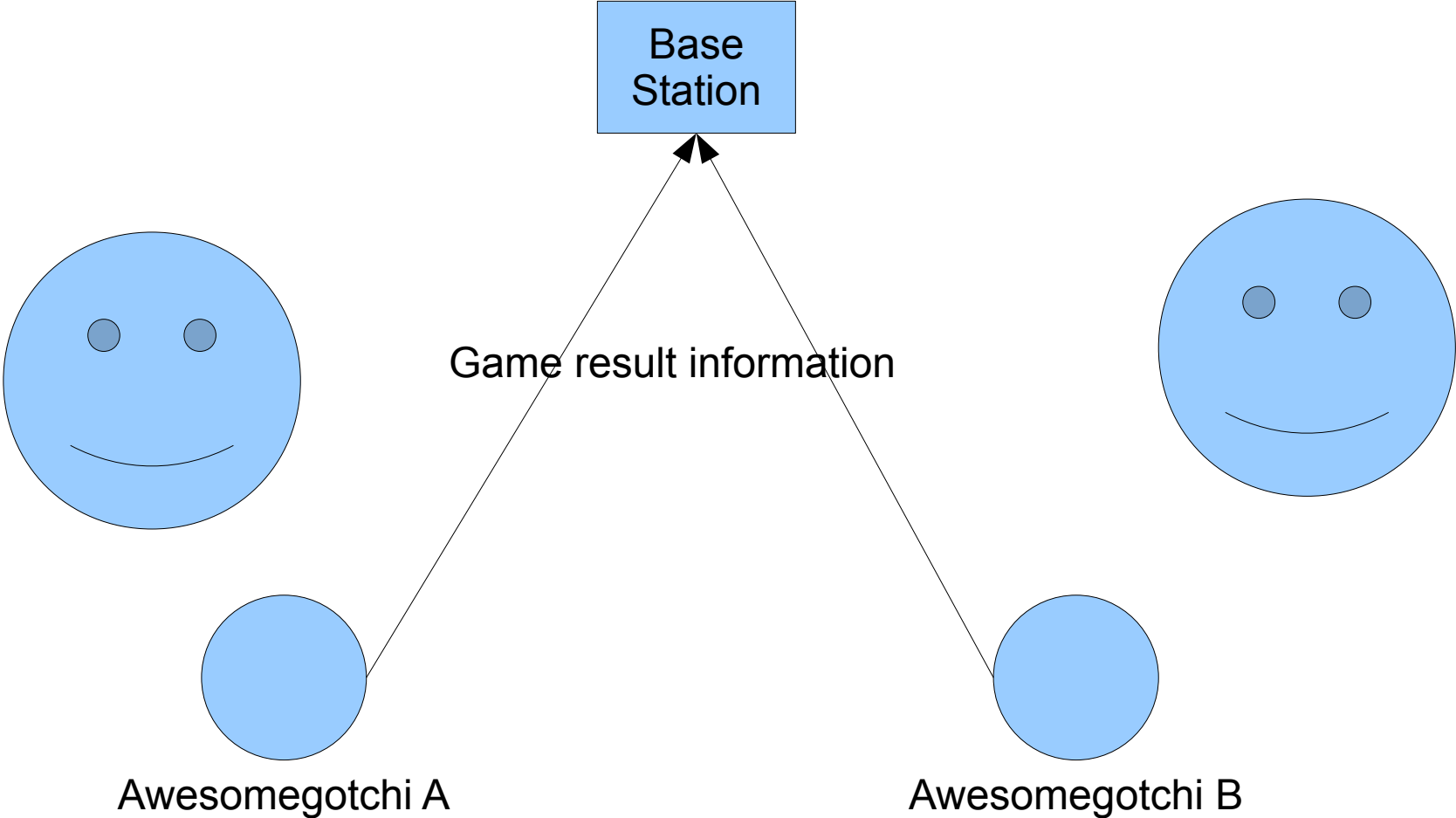
Game Start



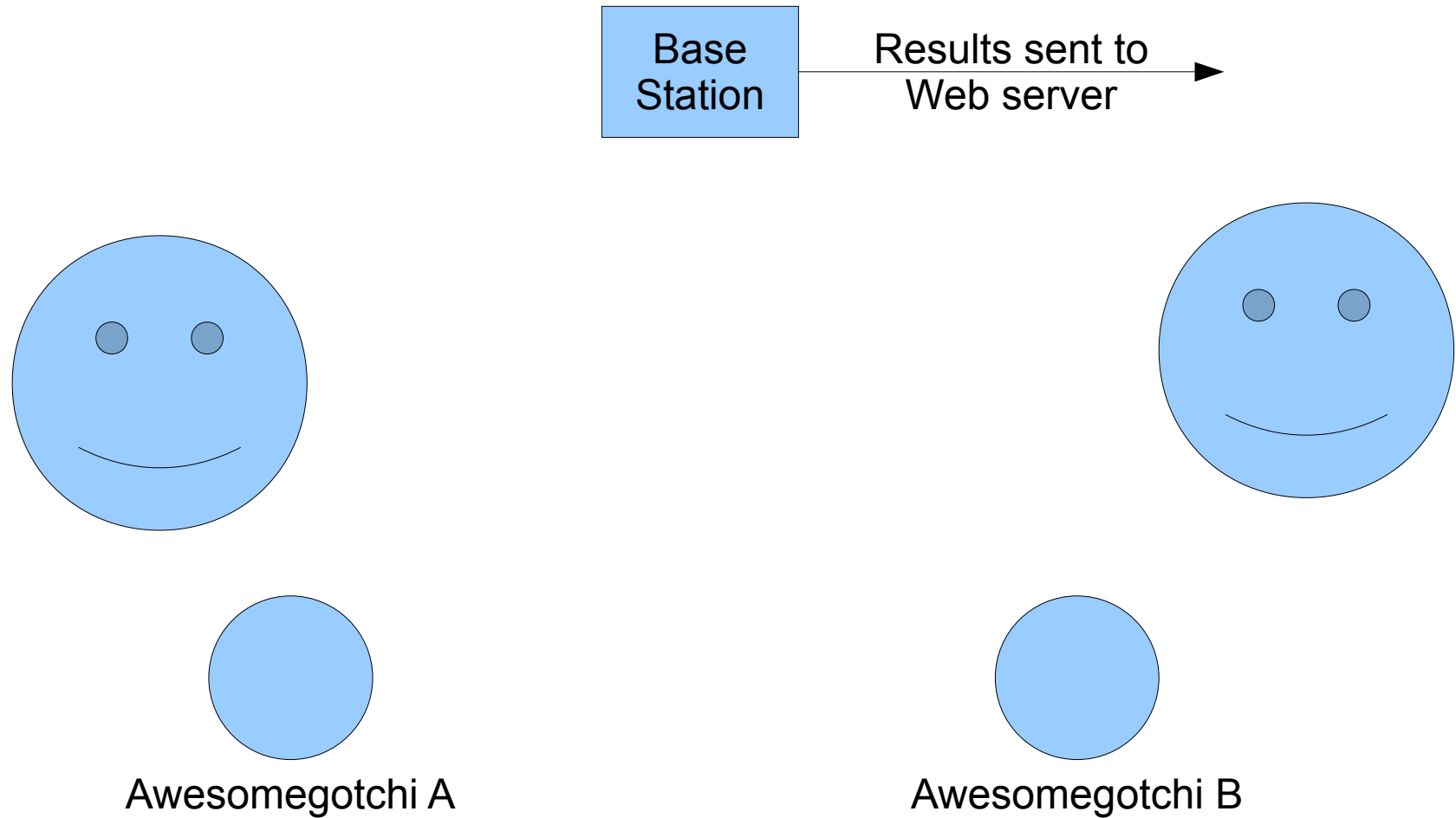
# Example Use



# Example Use



# Example Use



# Example Use

- This allows us to see that the users have met at a certain location and played a game together
- The users are provided with a leaderboard and result information
- Multiple base stations can be set up in order to allow location-specific gaming
- Gaming can be done without a base station, but results are not immediately uploaded

# Networking

# Requirements

- Reliable Transmission
- Two-Level Device Discovery
  - Base station discovery and association
  - Local device discovery

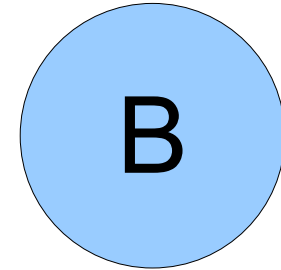
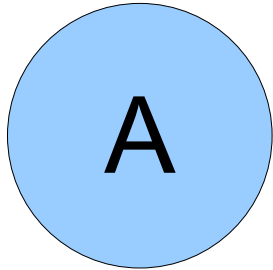
# Reliable Transmission

- Standard wireless system is not reliable
  - Interference from other devices
  - Shadowing (obstacles) and fading (distance) from the environment
- 'Reliable' essentially means 'we know the packet has been delivered'

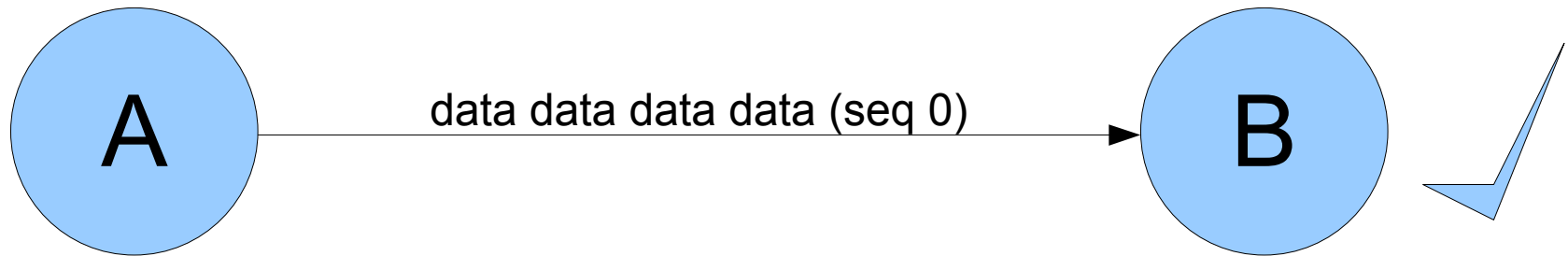
# Stop-And-Wait

- Very simple protocol
- Low memory and packet overhead
- Low bandwidth – but this is not generally a problem for embedded devices

# How Does It Work?



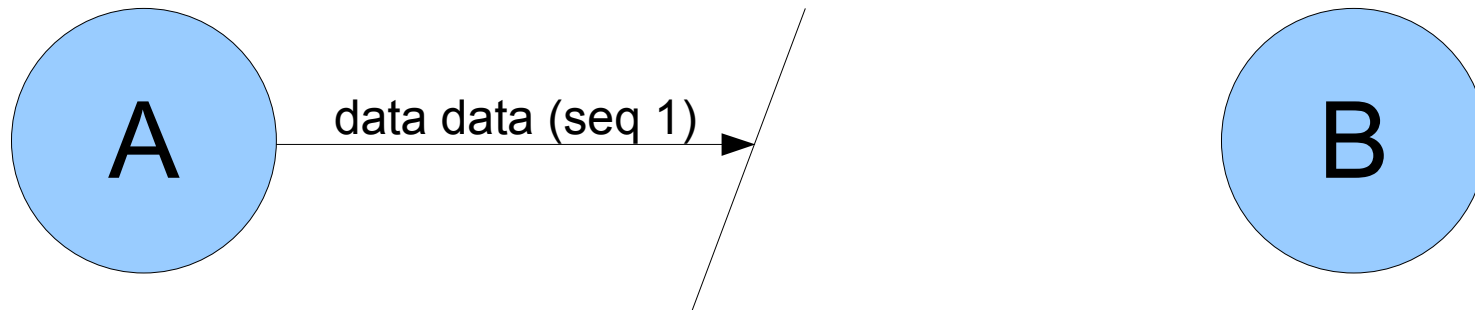
# How Does It Work?



# How Does It Work?

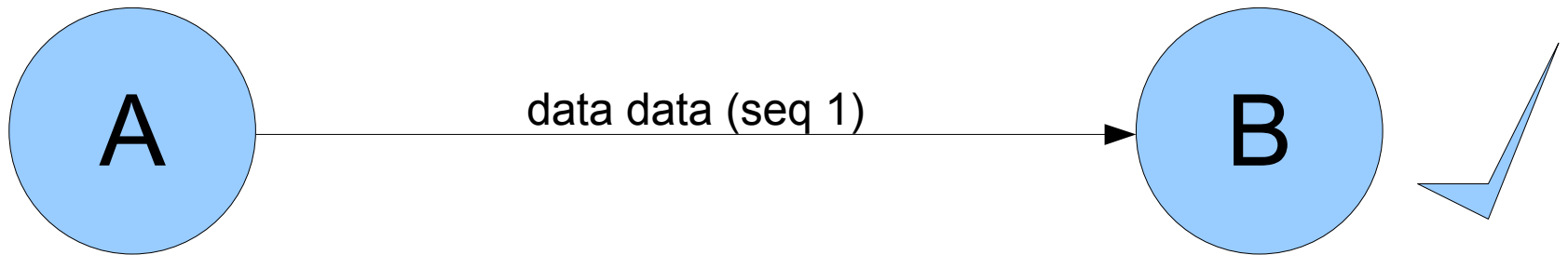


# How Does It Work?

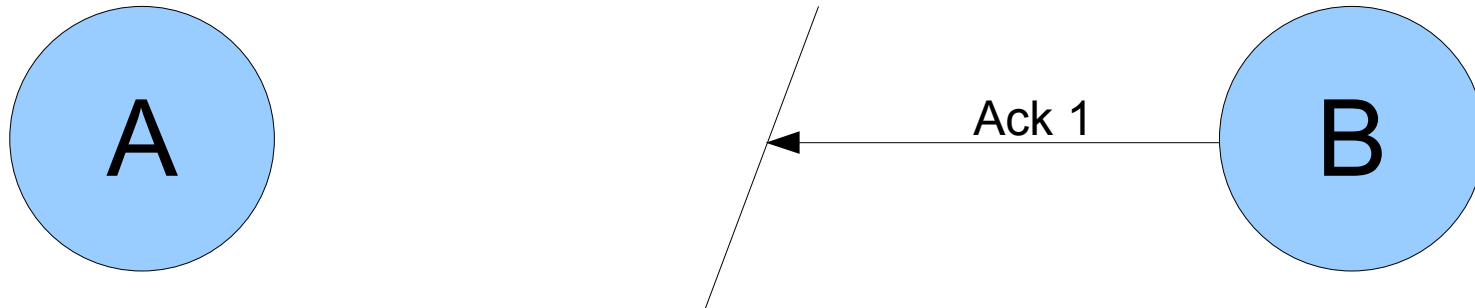


# How Does It Work?

...Some Time Later...

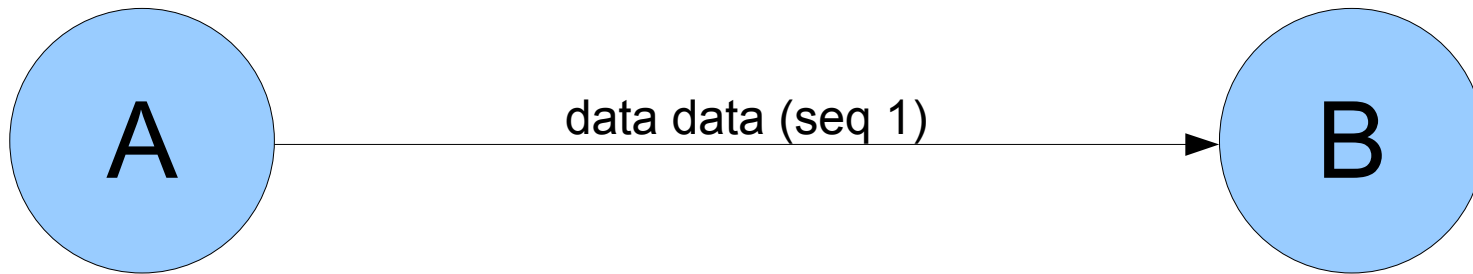


# How Does It Work?



# How Does It Work?

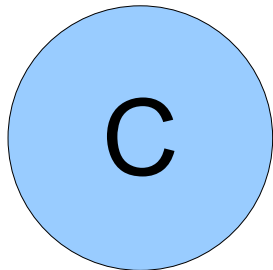
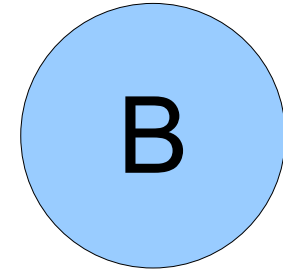
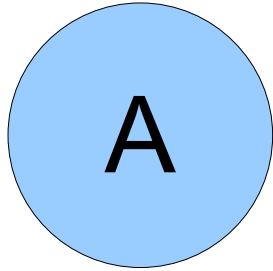
...Some Time Later...



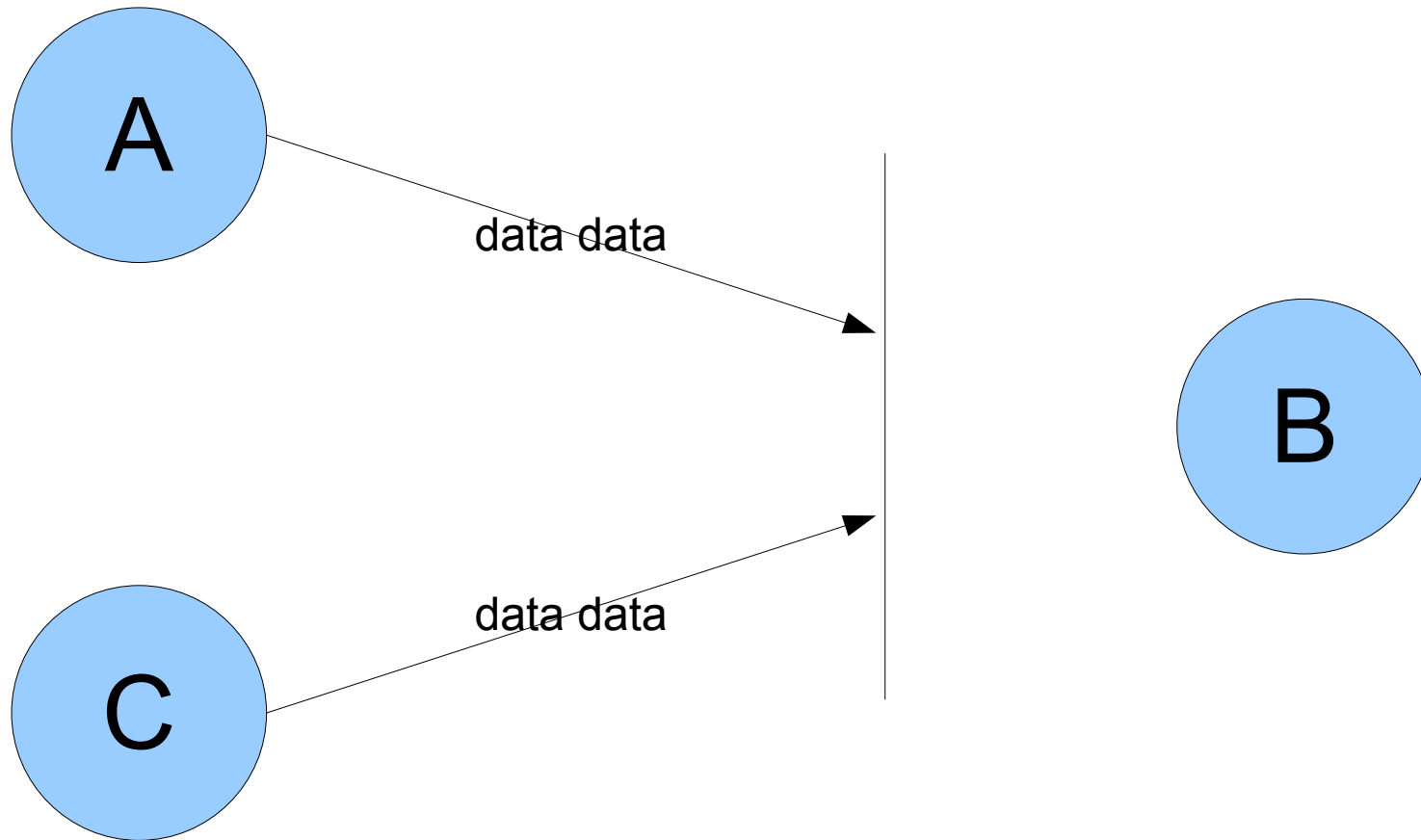
# How Does It Work?



# How Does It Work?

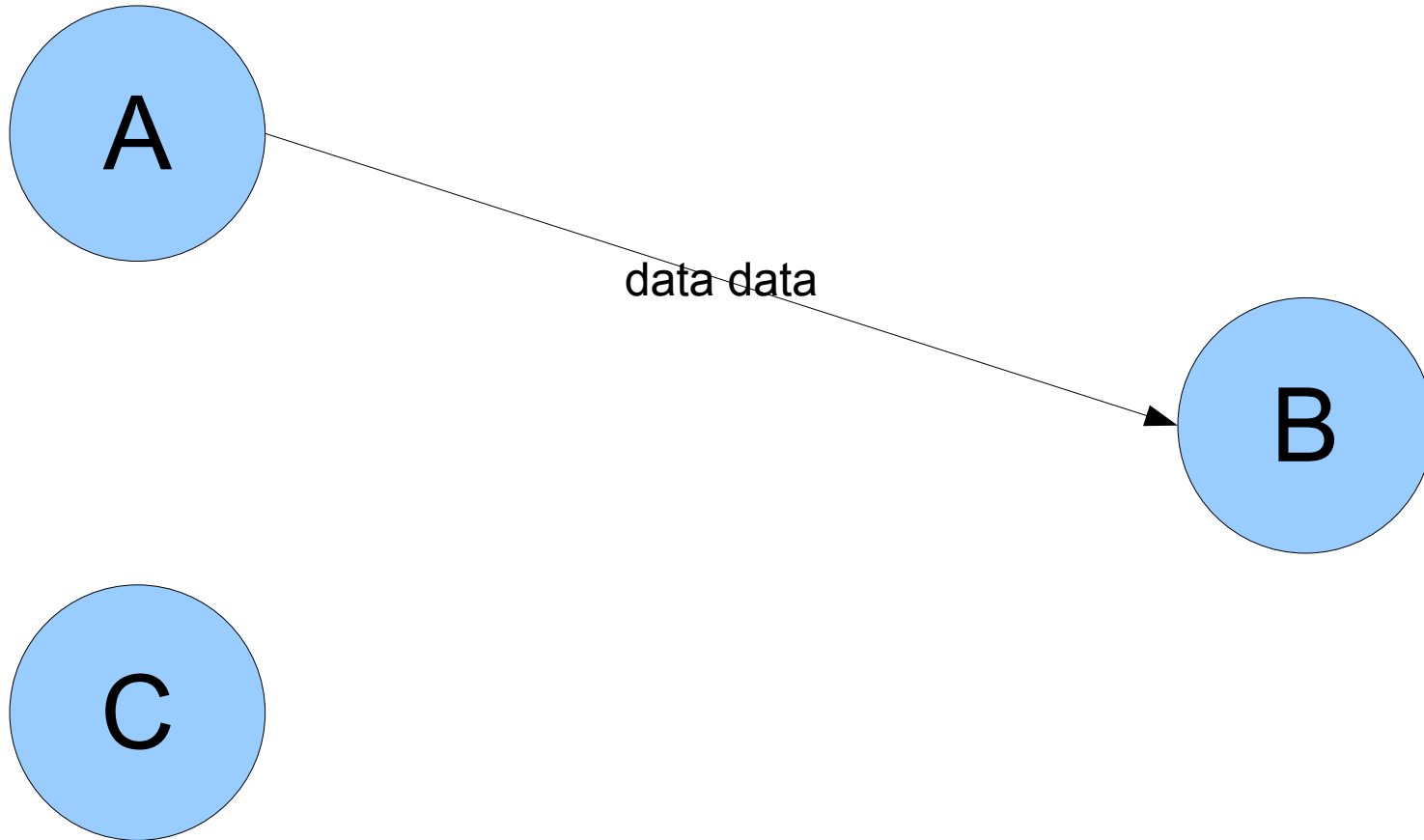


# How Does It Work?

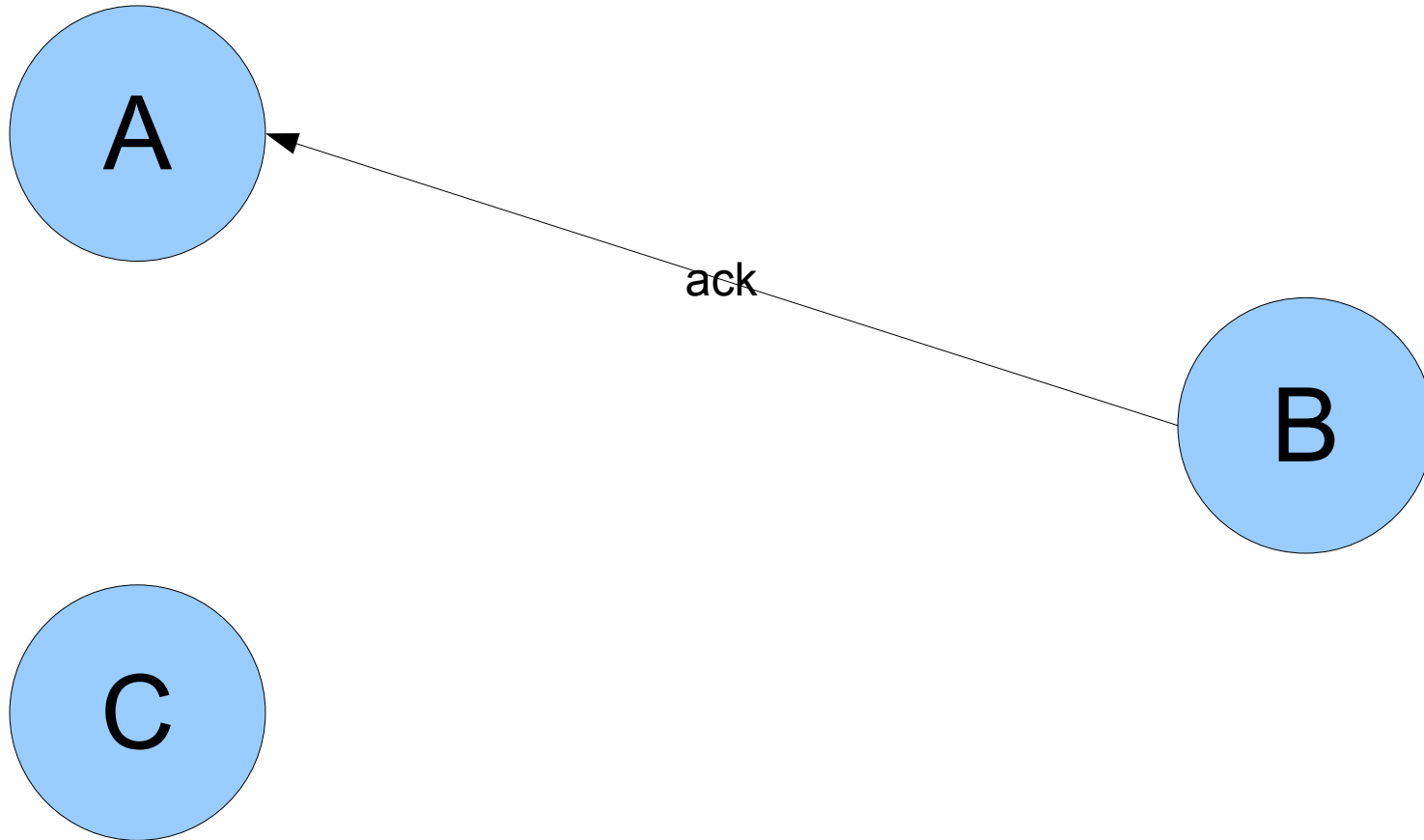


# How Does It Work?

...Some Time Later...

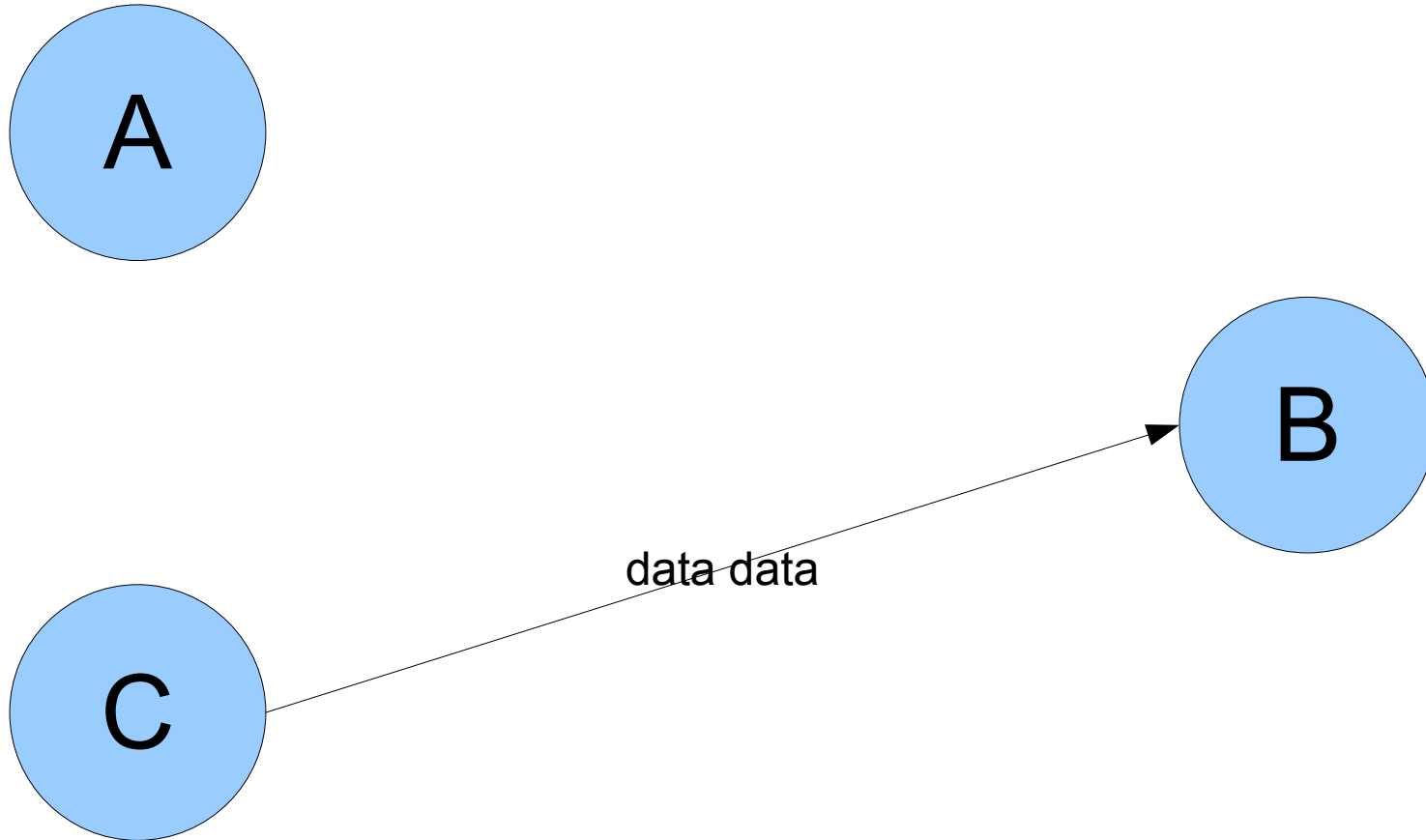


# How Does It Work?



# How Does It Work?

...Some Time Later...



# How Does It Work?

- Connections need to be stateful
  - Each device pair takes about 11 bytes
- If a connection becomes desynchronised (e.g. device reset) both devices reset their state
- Device-specific backoff based on device address

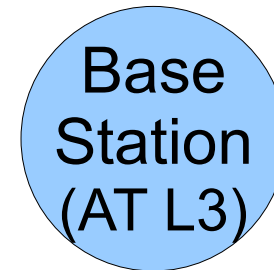
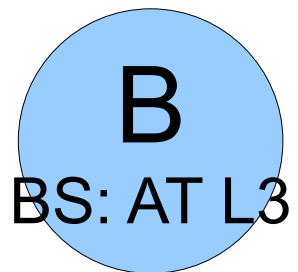
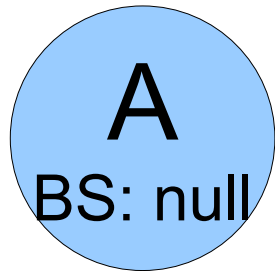
# Device Discovery

- Two-stage
  - 1) First, discover nearby base stations – closest base station is the one with the smallest hop count (assumes hops take equal time)
  - 2) Second, broadcast request for devices on same base station
    - Multiple beacons can be sent in order to ensure all devices are discovered
- Device addresses and 'tags' provided
- Tag is 14 bytes of data (used for sending names in our application)
- 14 bytes because screens are 14 chars wide

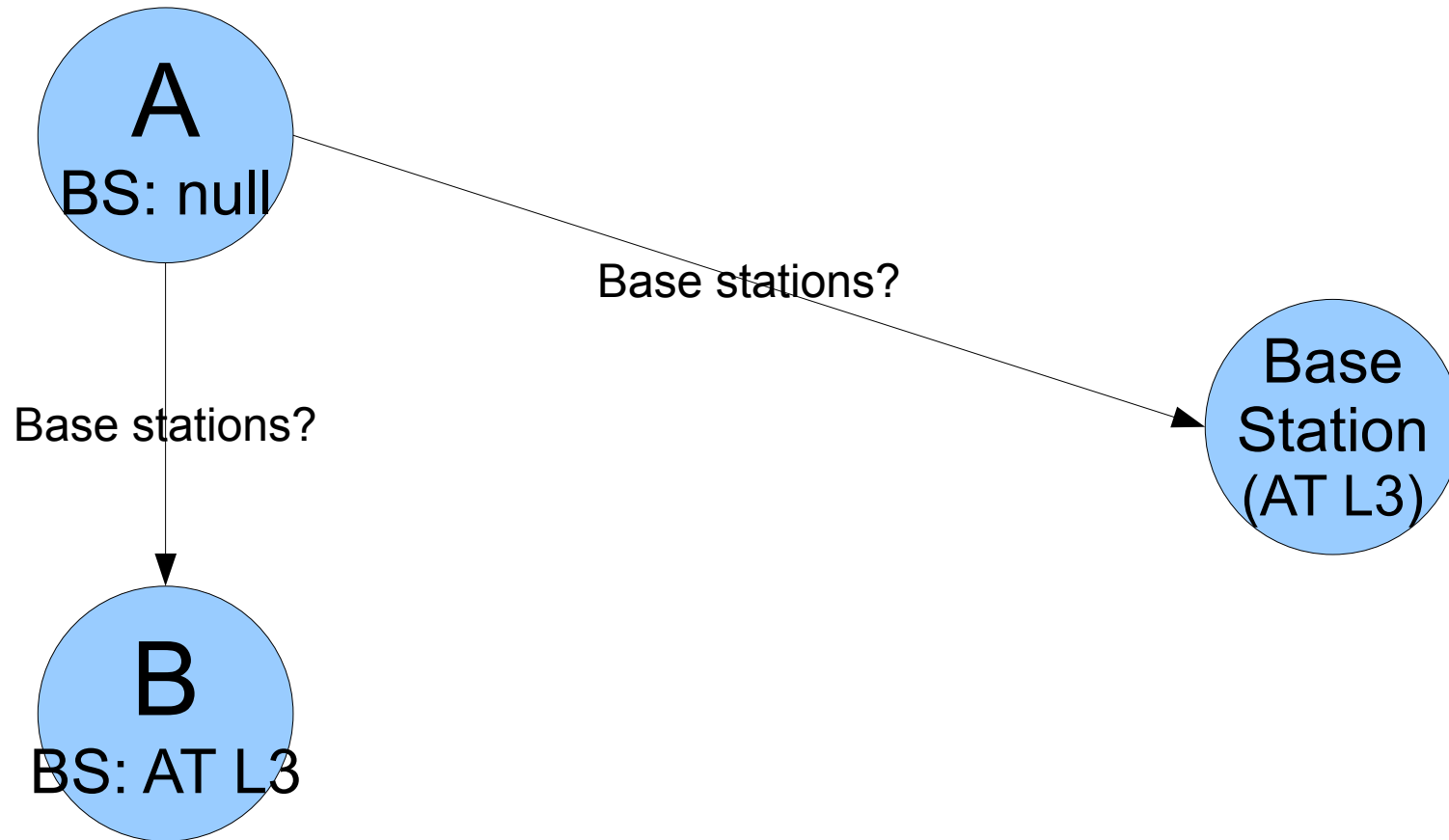
# Device Discovery

Discovery Dispatch	Packet Type	Data
Used to identify Discovery packets	Determines how packet is processed and what data field contains	Contains basestation address and/or data tag of sender
1 byte	1 byte	8 - 22 bytes

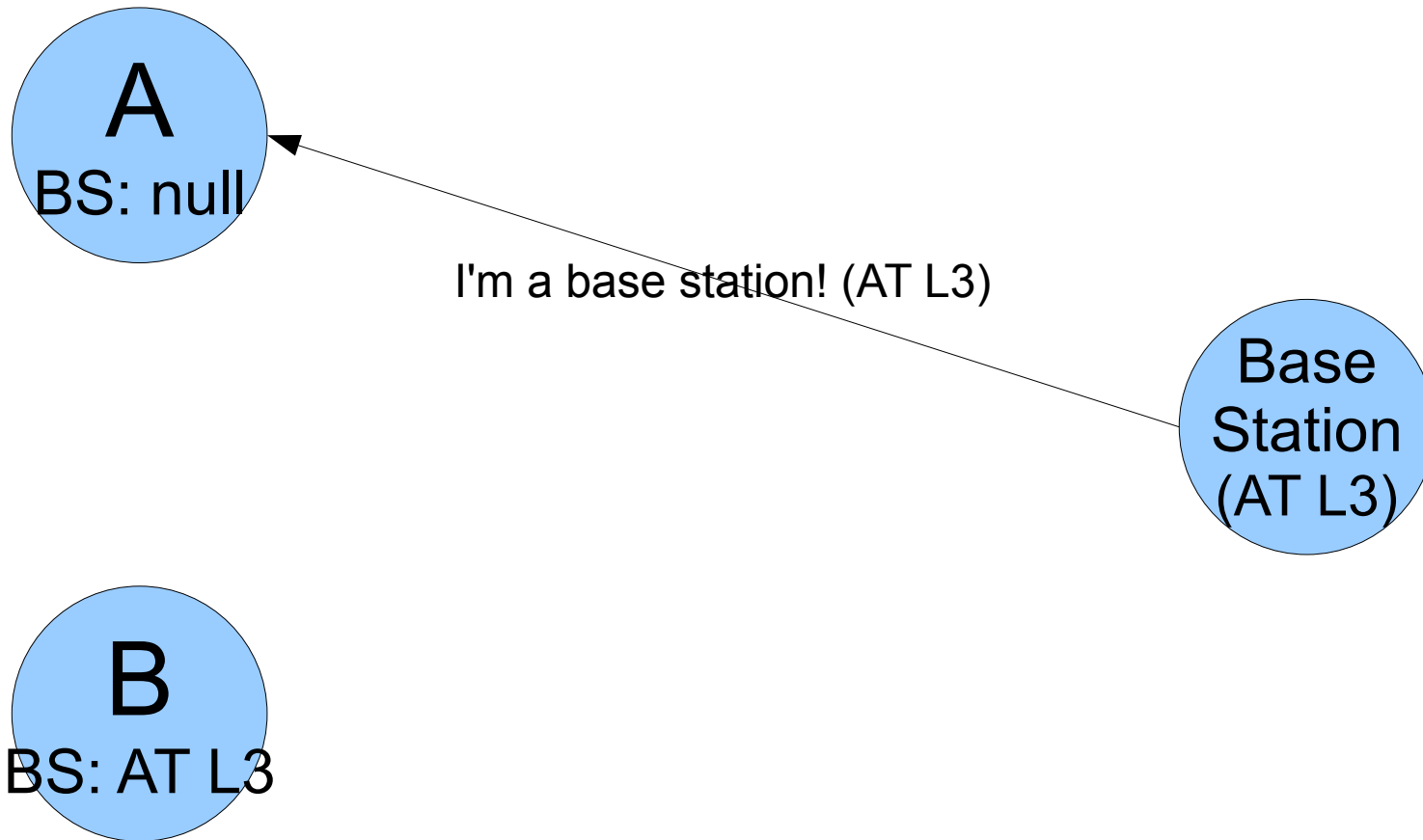
# Device Discovery



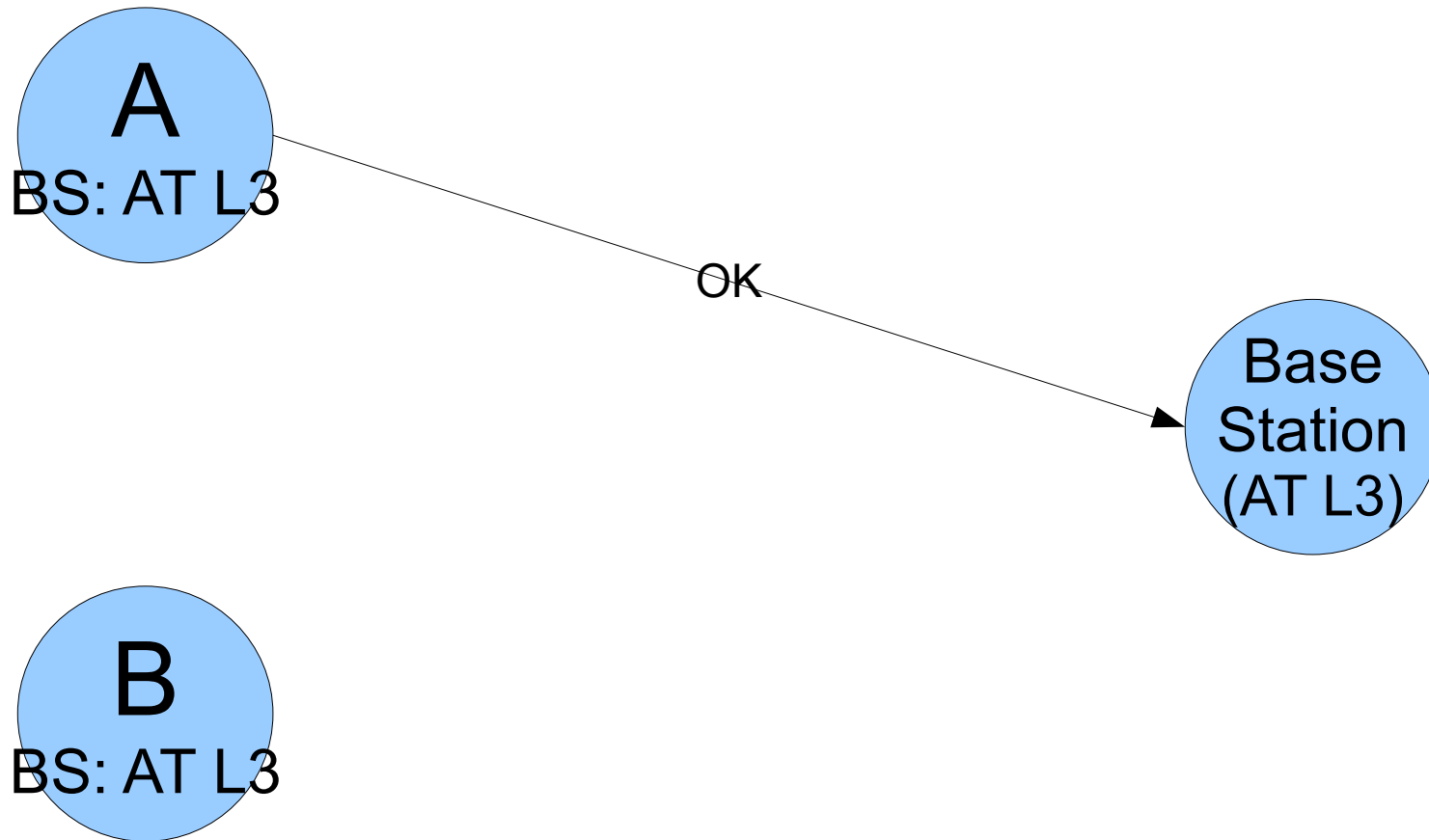
# Device Discovery



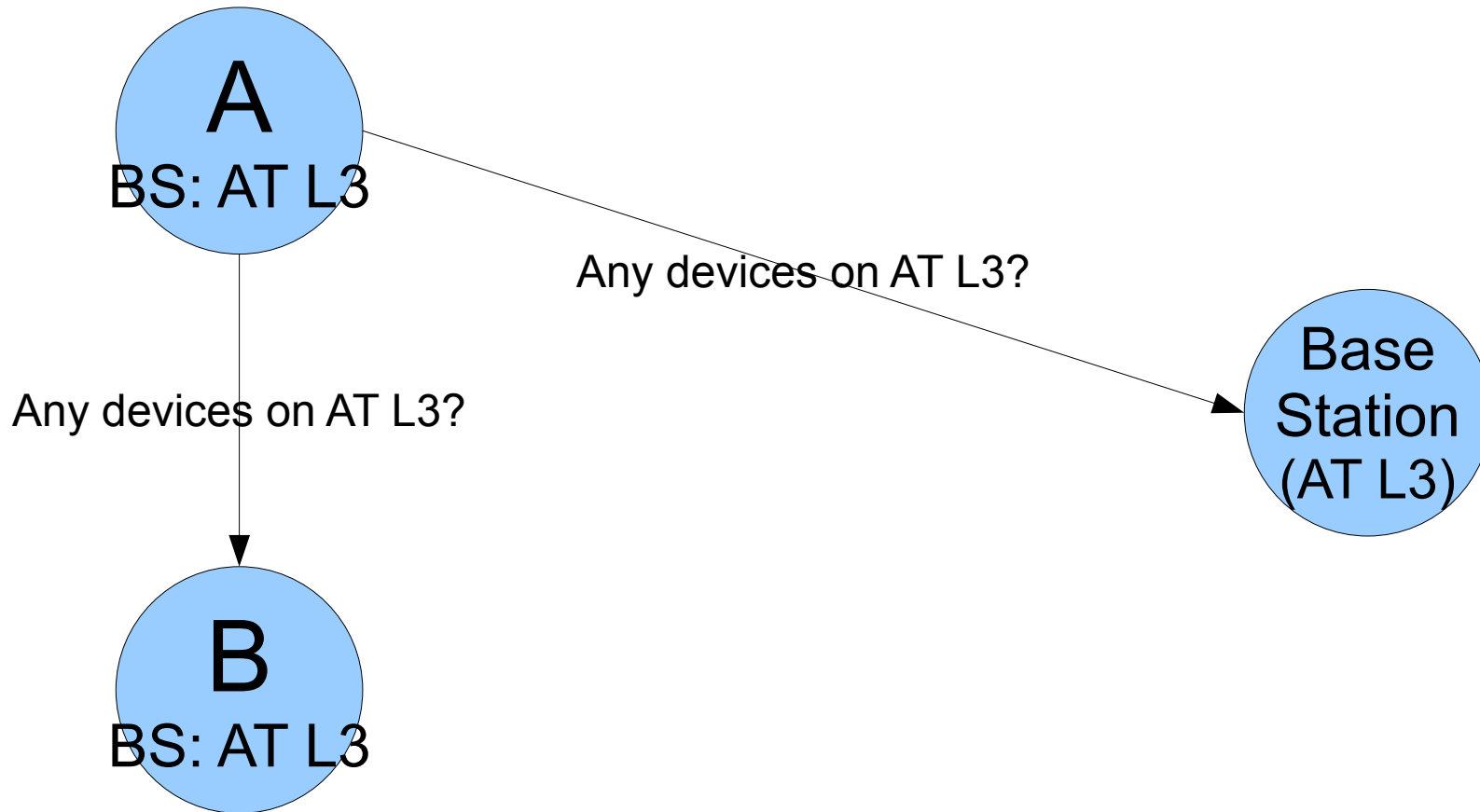
# Device Discovery



# Device Discovery



# Device Discovery



# Device Discovery

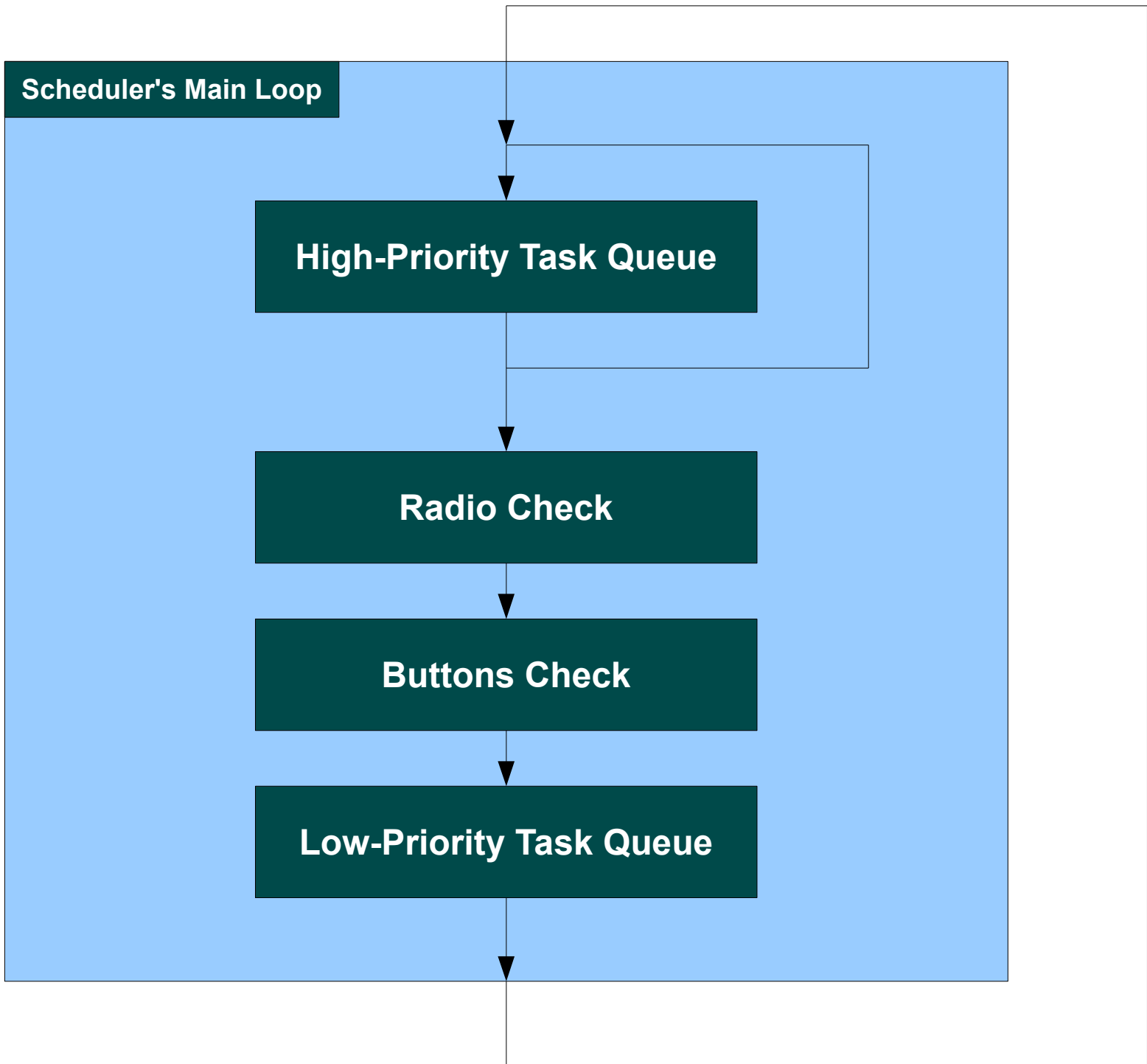


# Device Discovery

- A single basestation is associated with the Awesomegotchi
- Addresses of multiple found devices are stored

# The Scheduler

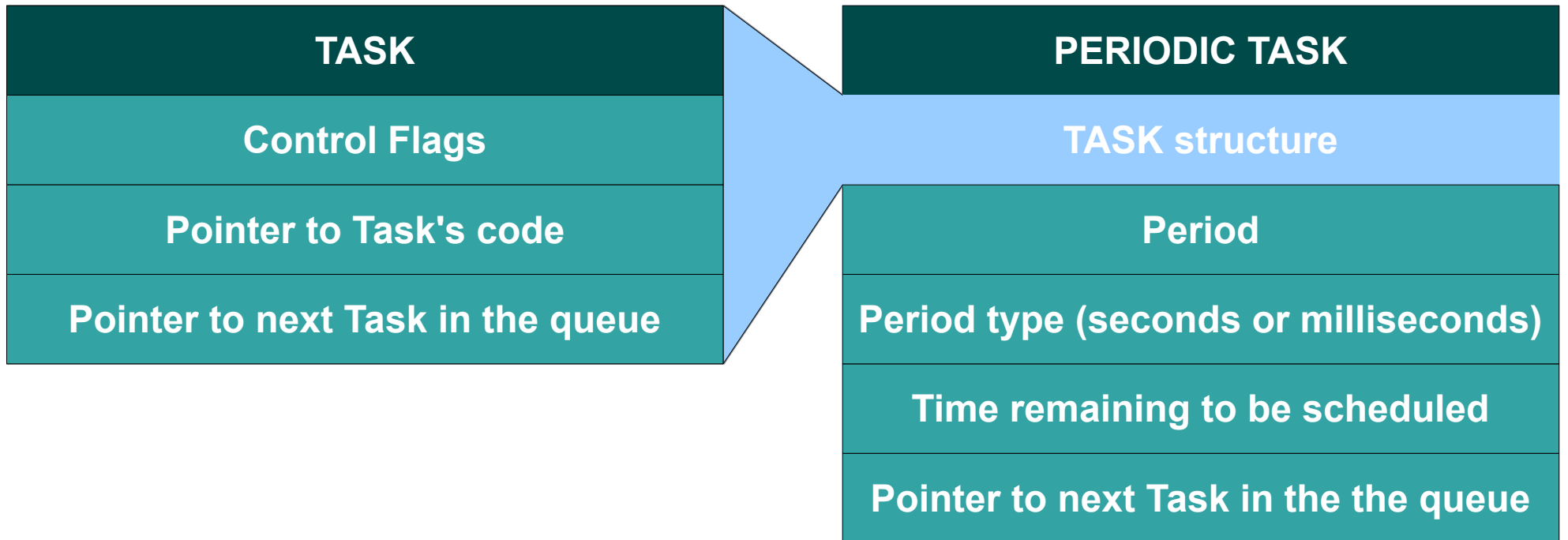
- Non-preemptive
- Two-level task priority
- Periodic and delayed scheduling
- Radio Check
- Buttons Check



# The Scheduler

- **Static tasks**
  - Created in the compile-time
  - Execution flow defined by tasks' internal states
- **Queues**
  - Defined as linked lists of tasks
- **Periodic and Delayed Tasks**
  - Two separate queues: second-based and millisecond-based
  - Scheduled when their time is passed

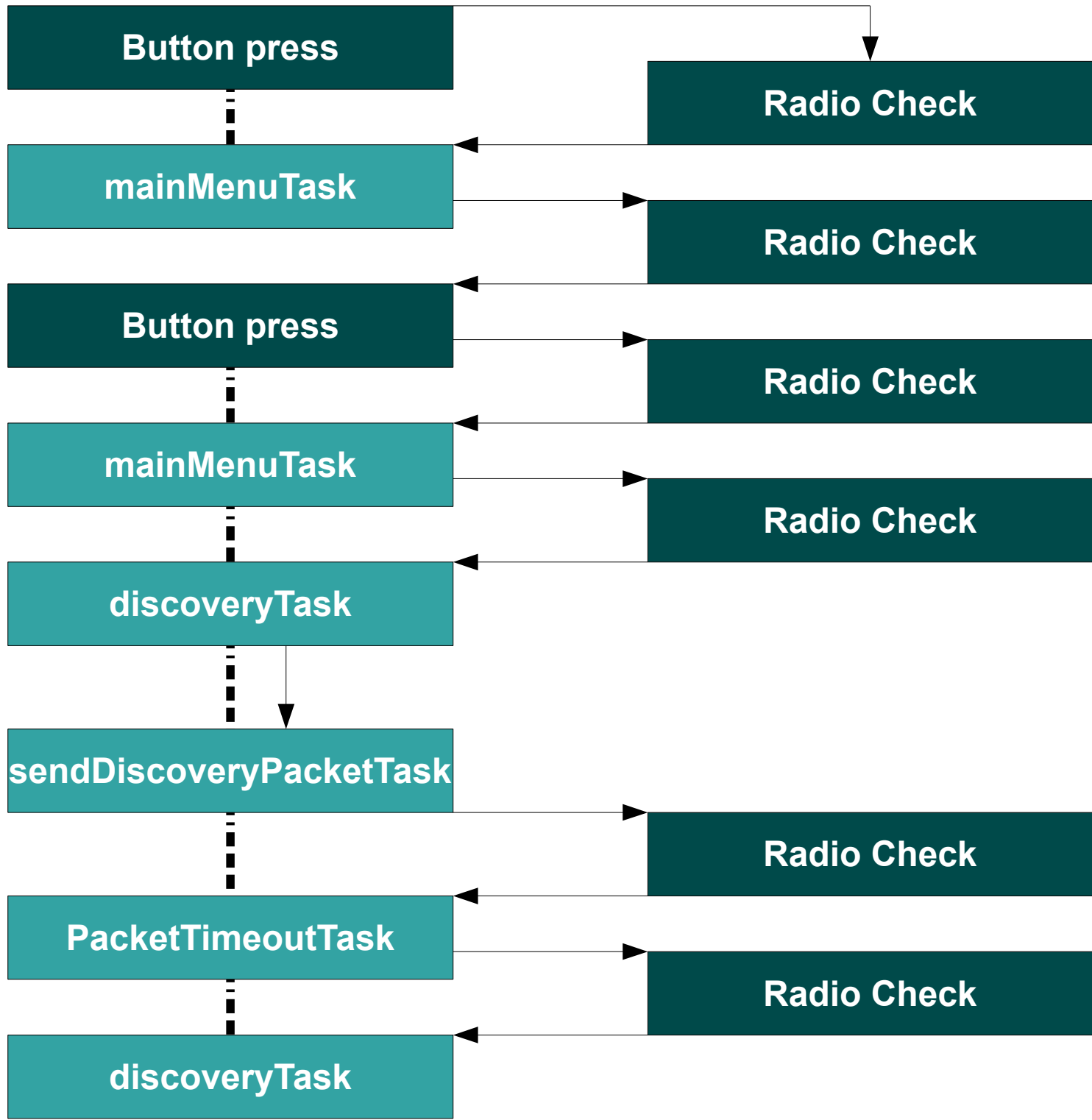
# Task Structure



# The Main Timer

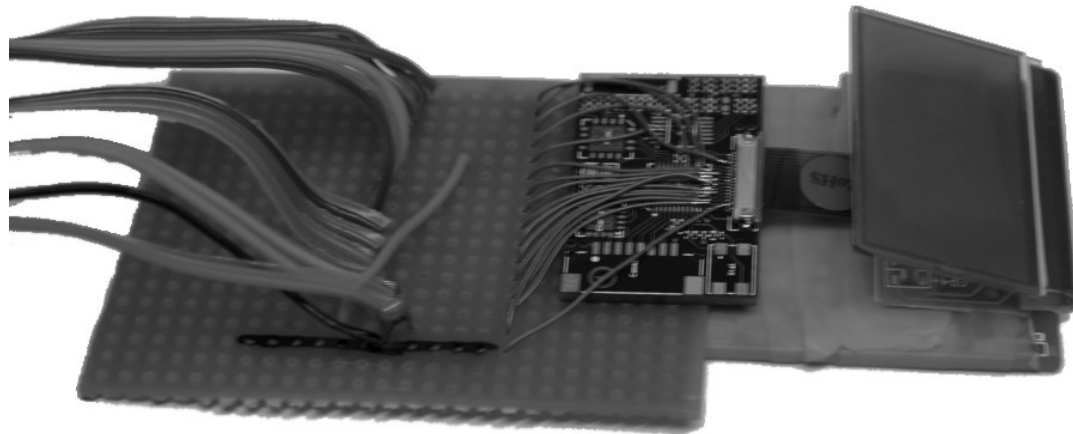
- Based on internal 32.768kHz oscillator
- ~1ms resolution
- Updates remaining time on the periodic/delayed tasks:
  - Every second for second-based tasks
  - When the task is due for millisecond-based tasks

**D  
I  
S  
C  
O  
V  
E  
R  
Y  
  
E  
X  
A  
M  
P  
L  
E**



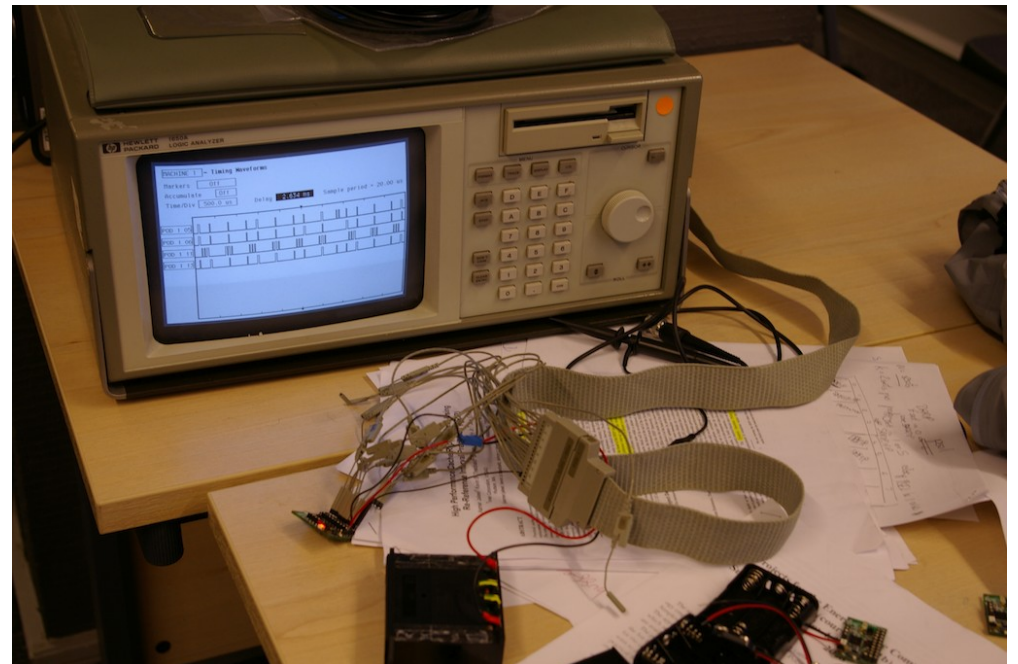
# Screens

- Parallel Interface – 8 bit bus with 5 control lines
- 64x128 pixels
- Allows graphic and live text display



# Screen Interface Development

- Required use of a logic analyzer to correctly set timings of pins



# Screen Interface

- 8 'lines' – each 8 bits wide
- Accessed as memory blocks with each 128 bytes controlling pixels along that line

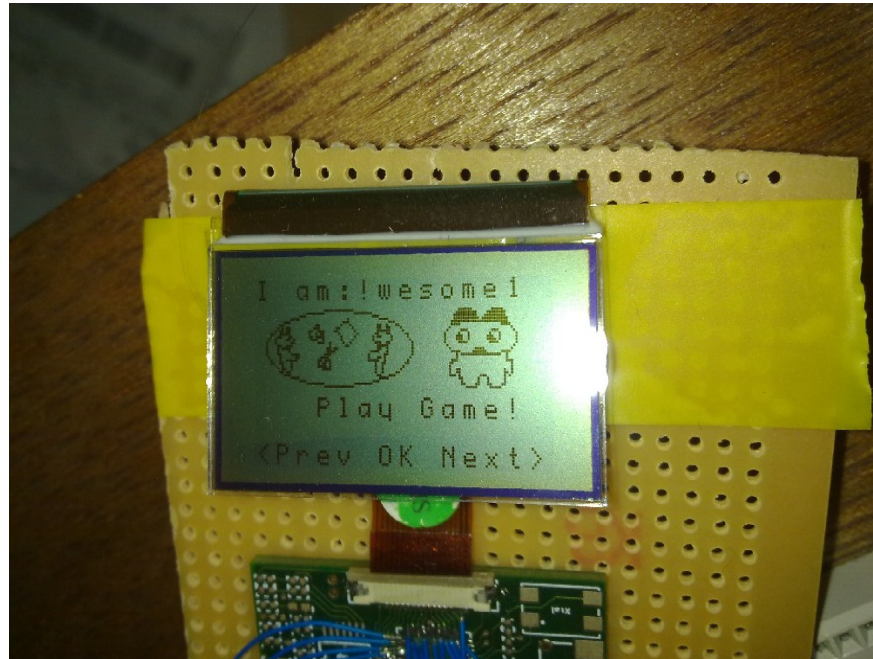


# Character Set & Graphics

- All ascii characters optimized for this screen
- 7 x 8 bits wide – fitting 14 on a line
  - Had to be rotated 90 to be displayed horizontally
- Can display bitmaps and RLE encoded graphics

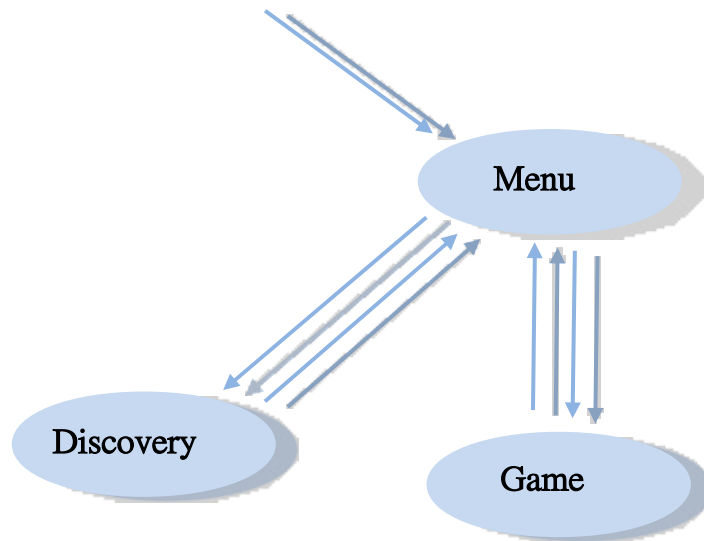
# Menu

- Menu runs as a 'Task'
- Easy to add more 'Programs' or 'Tasks'



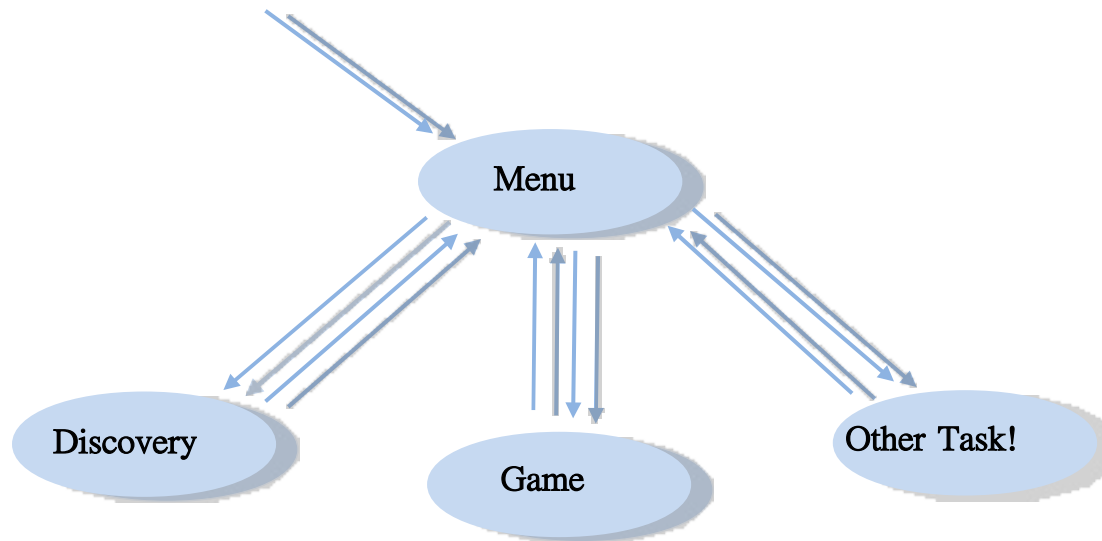
# Menu

- The current menu structures as an FSM

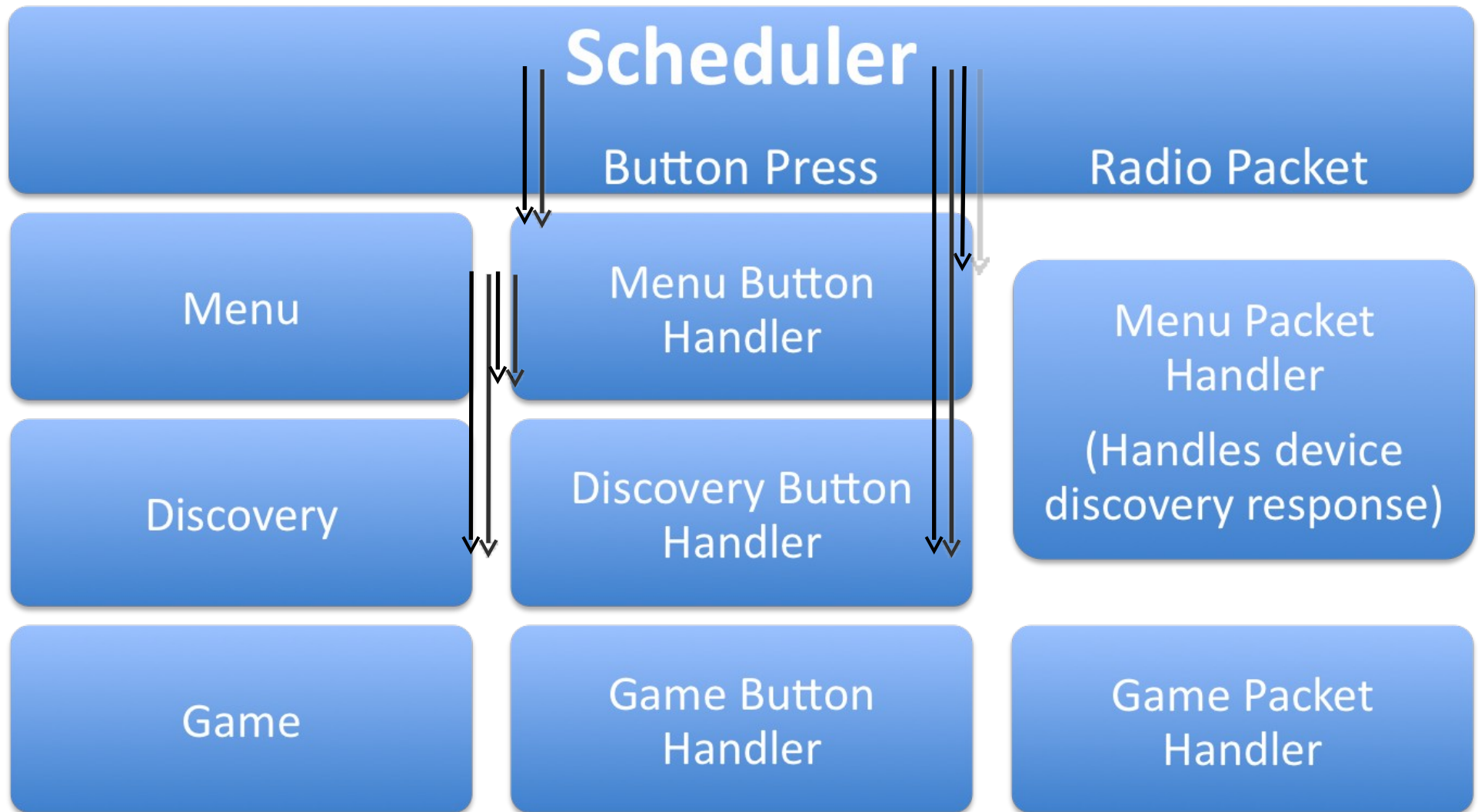


# Menu

- But it is easy to add more applications!

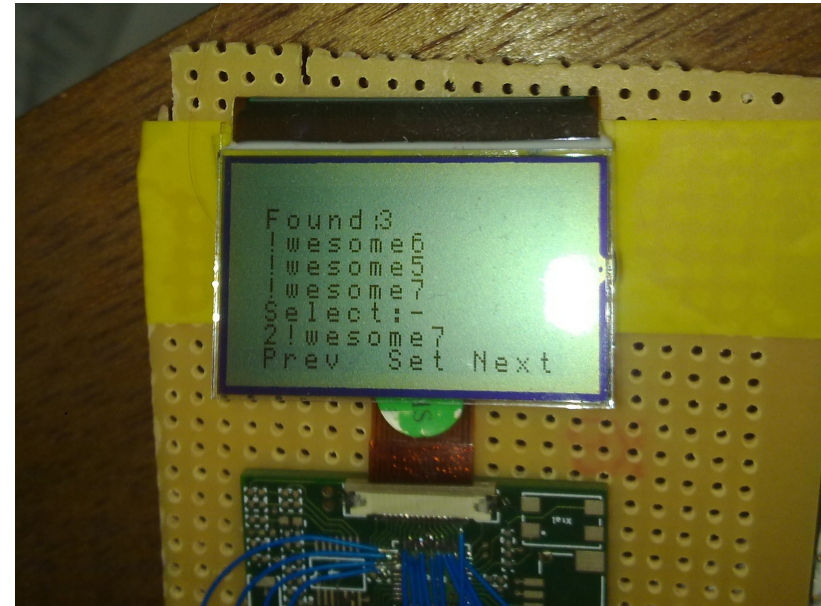
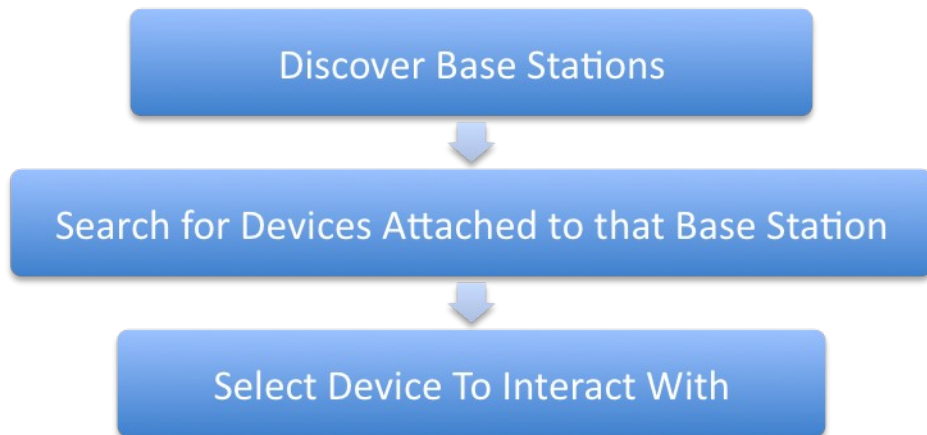


# Application Functional Units



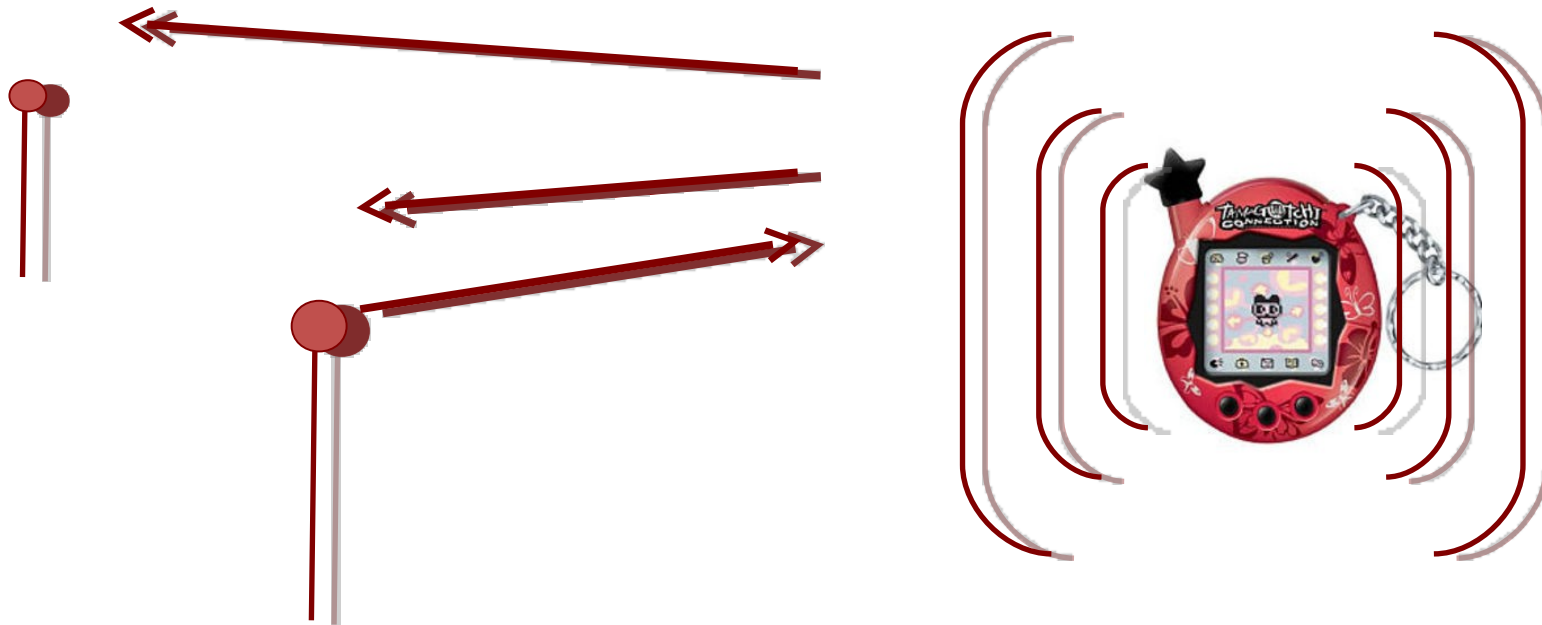
# Device Discovery Interface

- Detects nearest base station
- Allows user to select device to game with



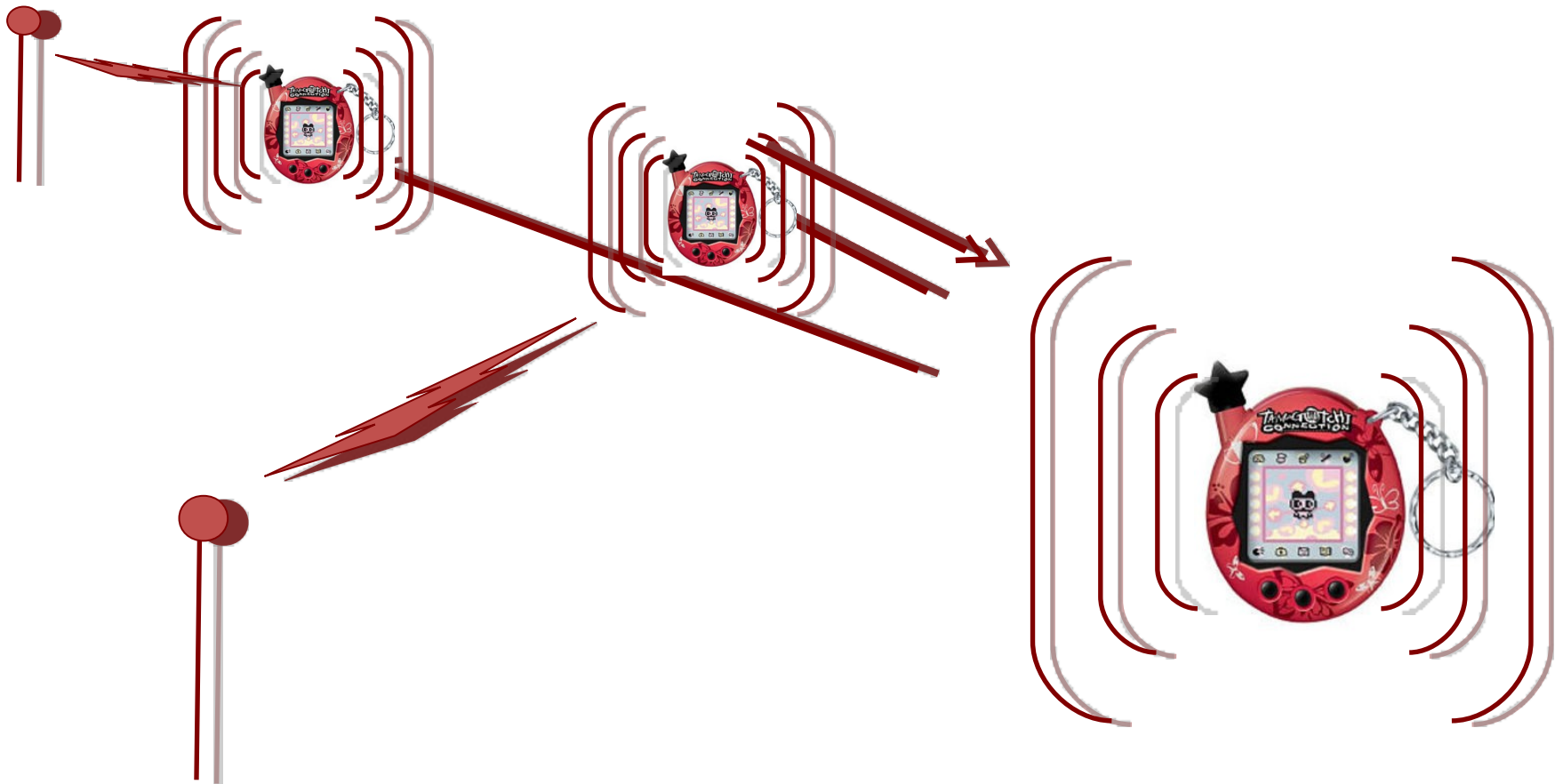
# Device Discovery

- Sends a 'base station' broadcast, to which only base stations respond
  - The first base station to respond is selected



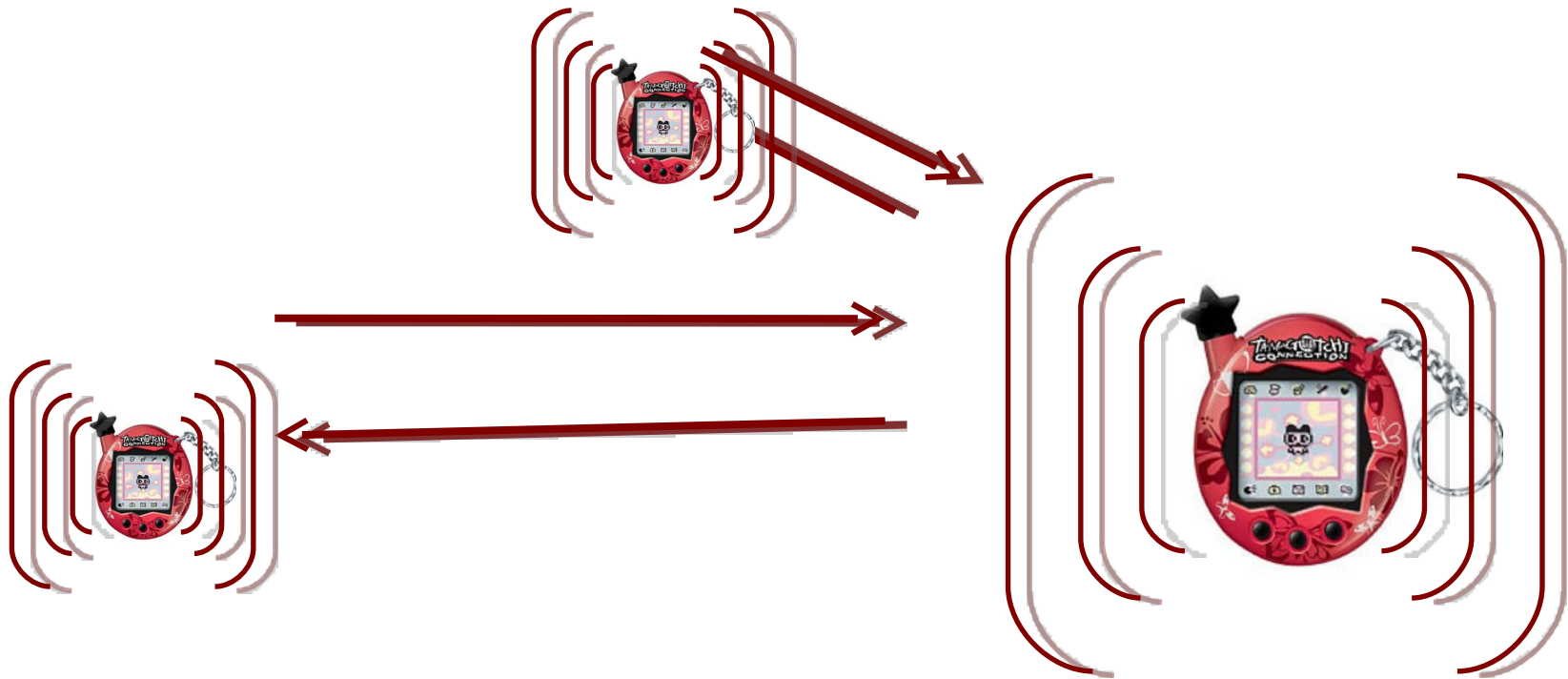
# Device Discovery

- Then broadcasts for devices associated with that base station

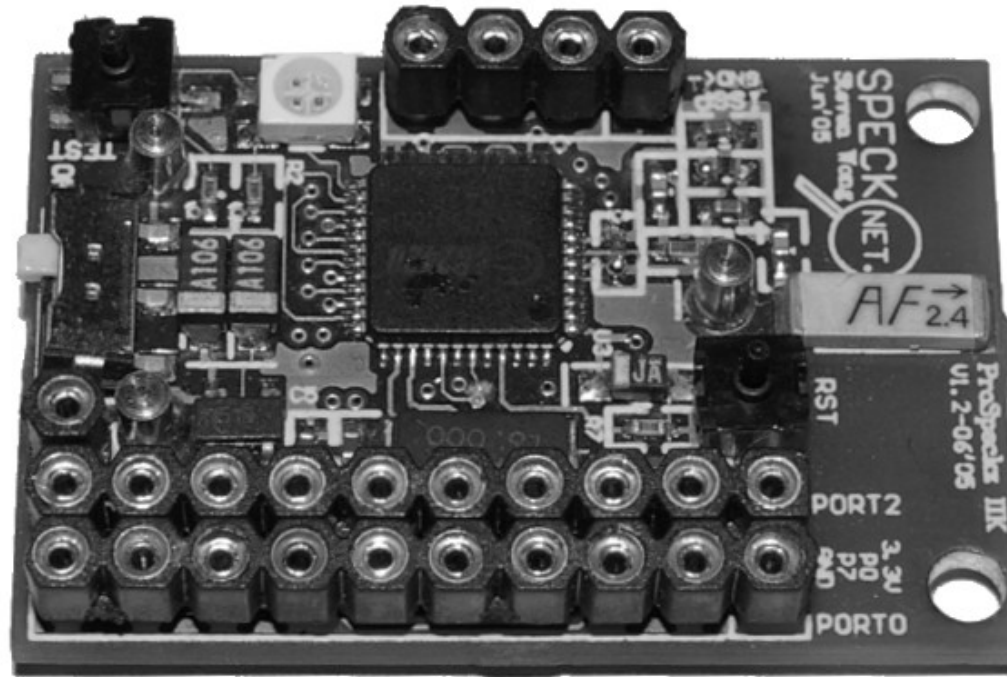


# Device Discovery

- If no base station is found it searches for all nearby devices

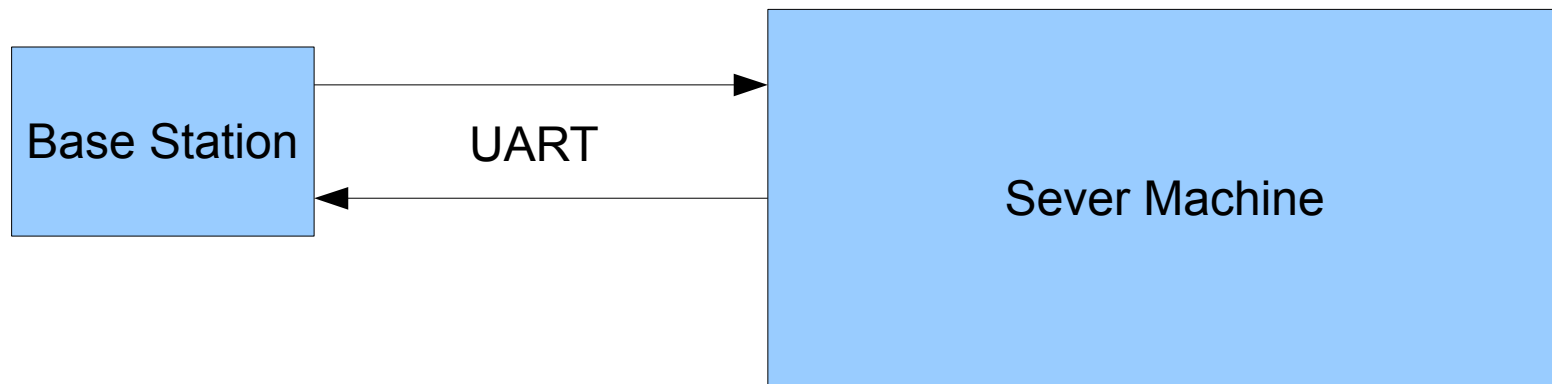


# Base Station



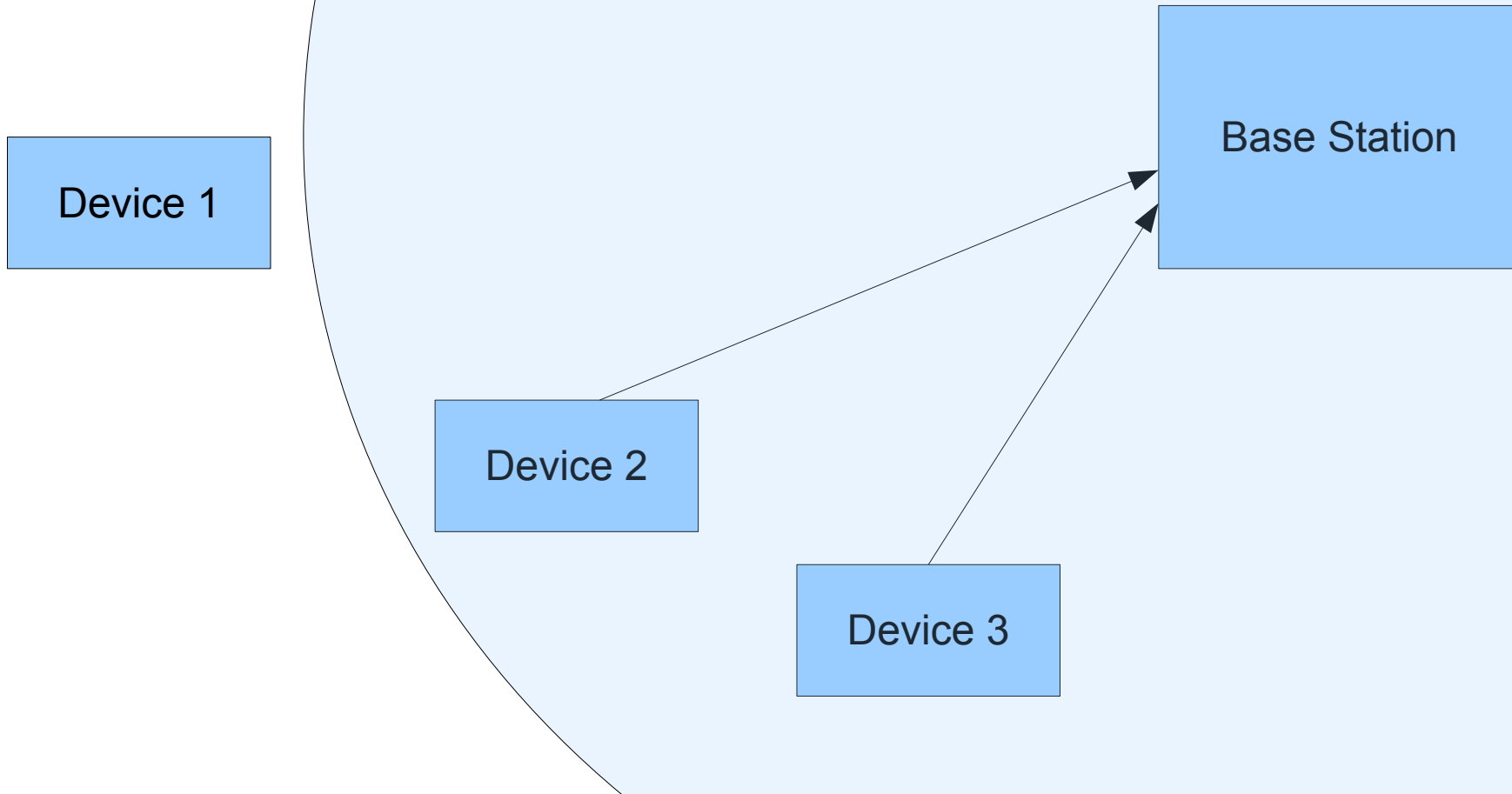
# How it works

Communication with the server machine over a Serial interface using UART (Universal Asynchronous Receiver Transmitter).



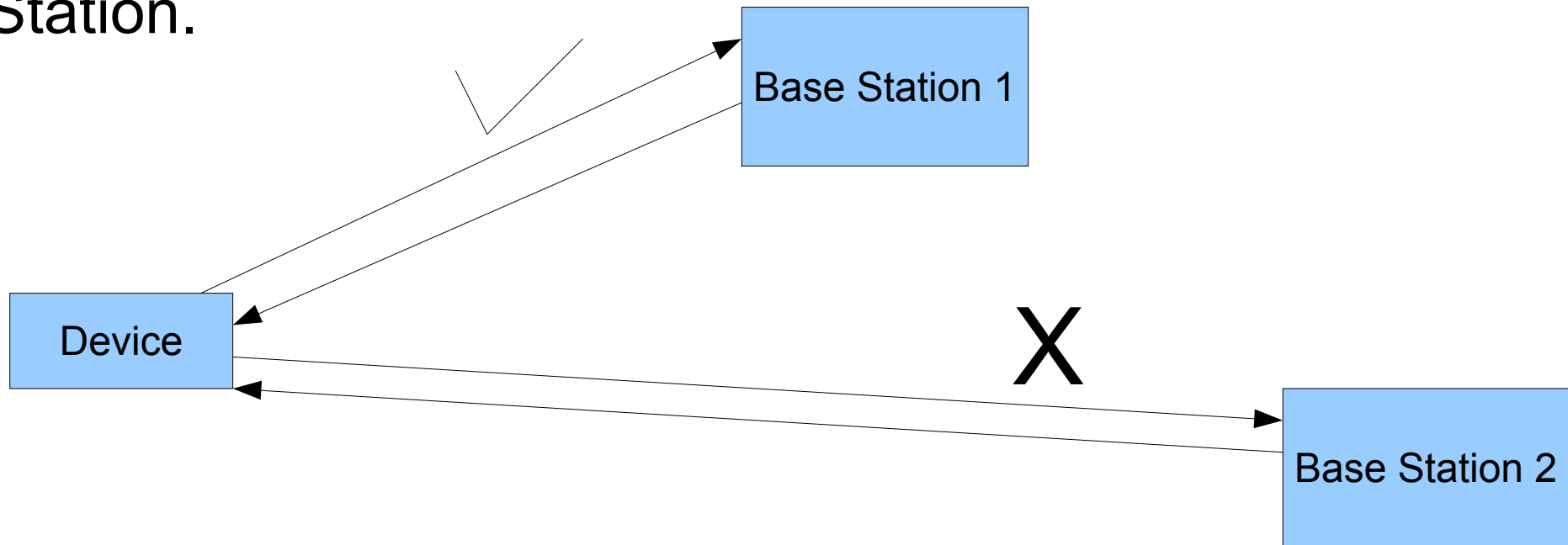
# How it works

Devices associate with the Base Station whenever they are within its range.



# Association

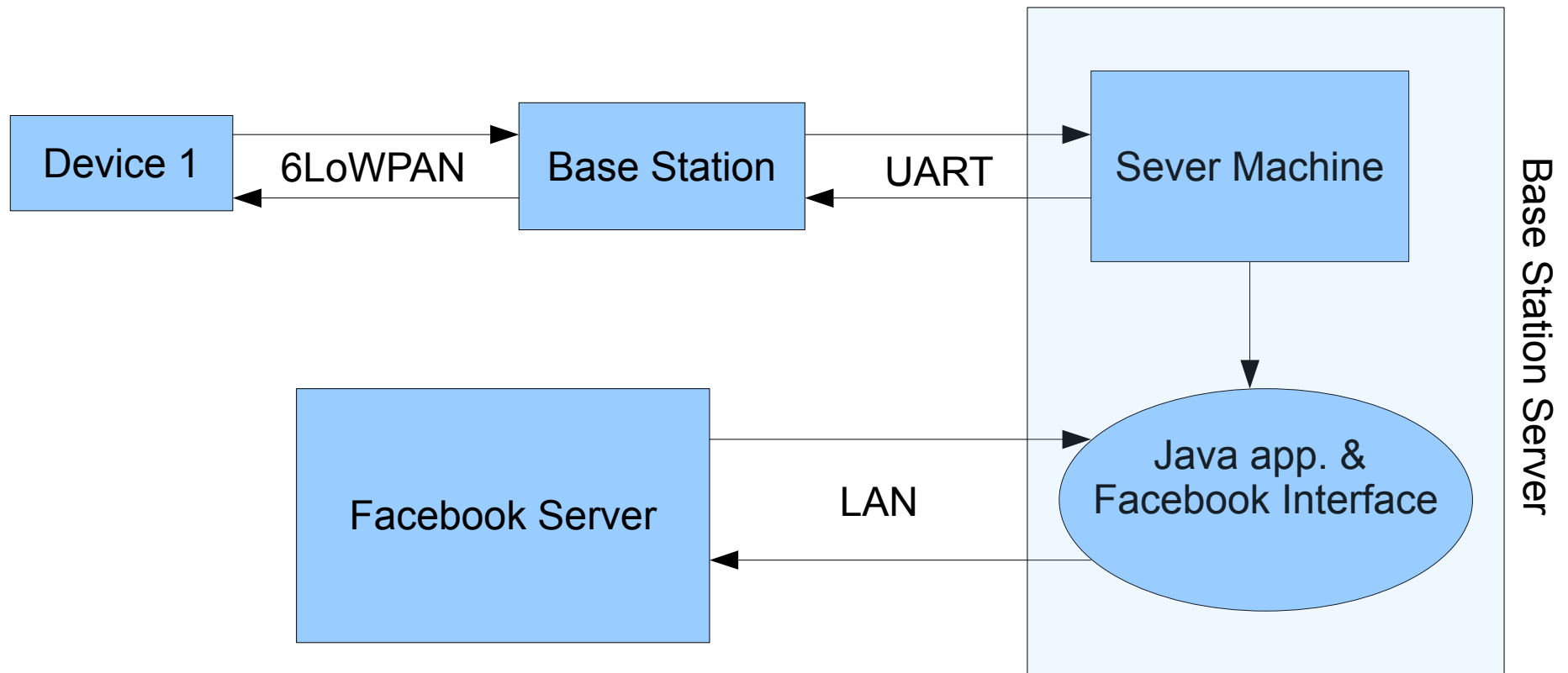
A device sends a request to associate with a Base Station.



The first Base Station to respond establishes a connection

# How it works

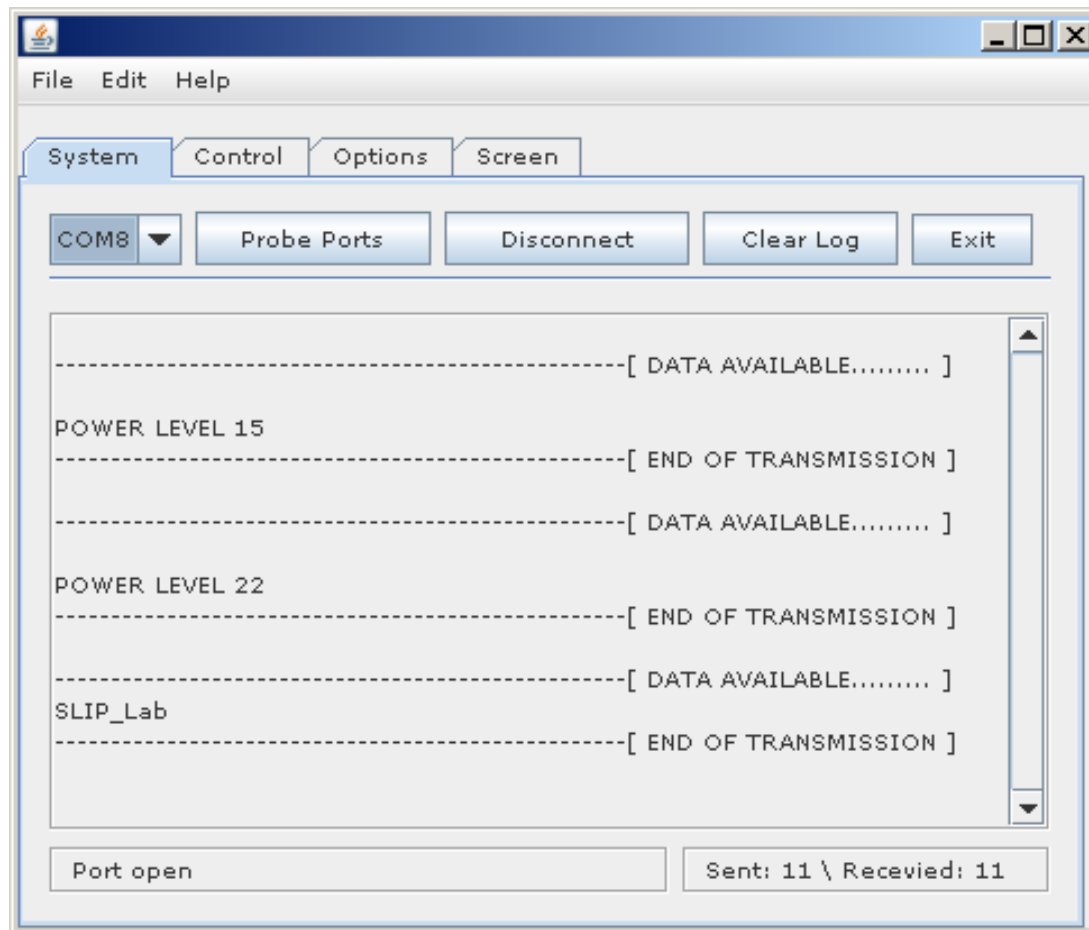
Information is processed by a Java application and forwarded to the Facebook server upon request.



# Java Application

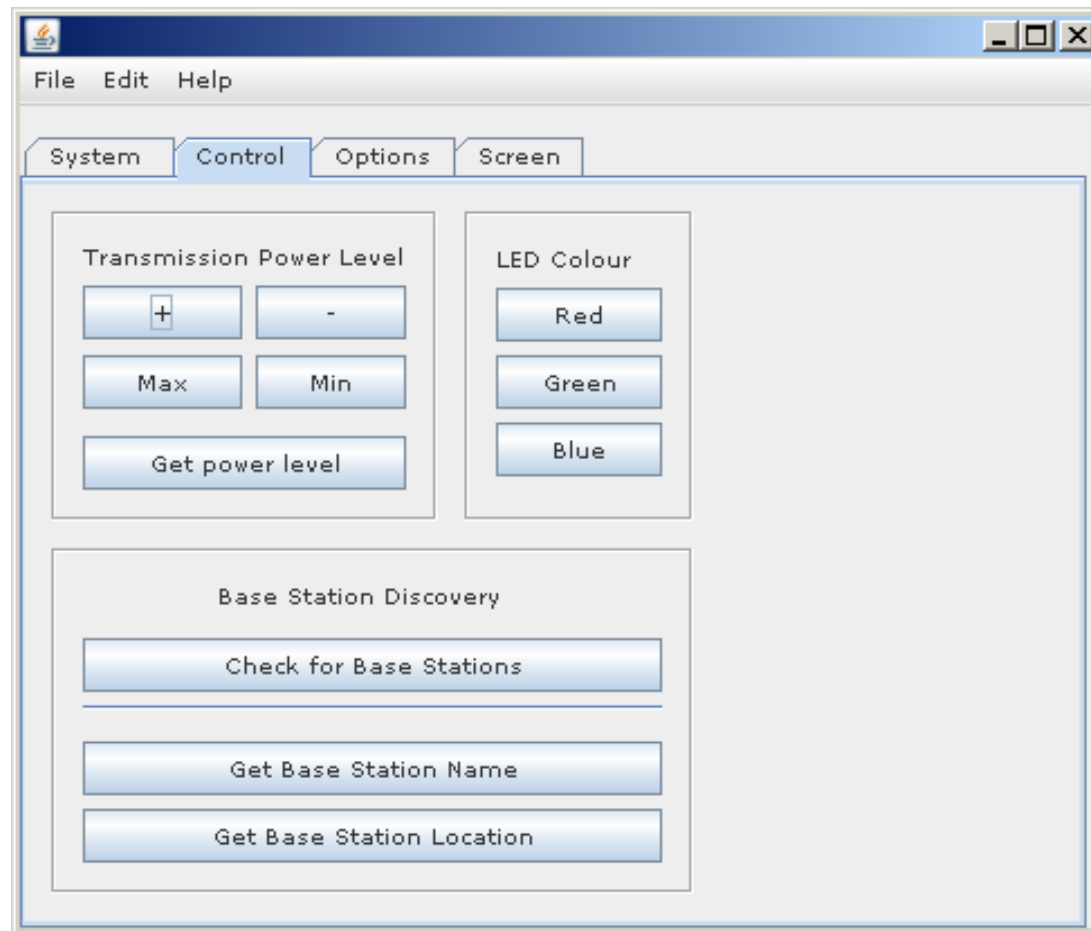
- Displays incoming packet content to aid debugging
- Controls Base Station device connected via Serial interface
- Checks and validates incoming packets
- Changes Base Station transmission power levels
- Keeps track of Base Station address and location
- Binds to Facebook interface

# Java Application : System Log

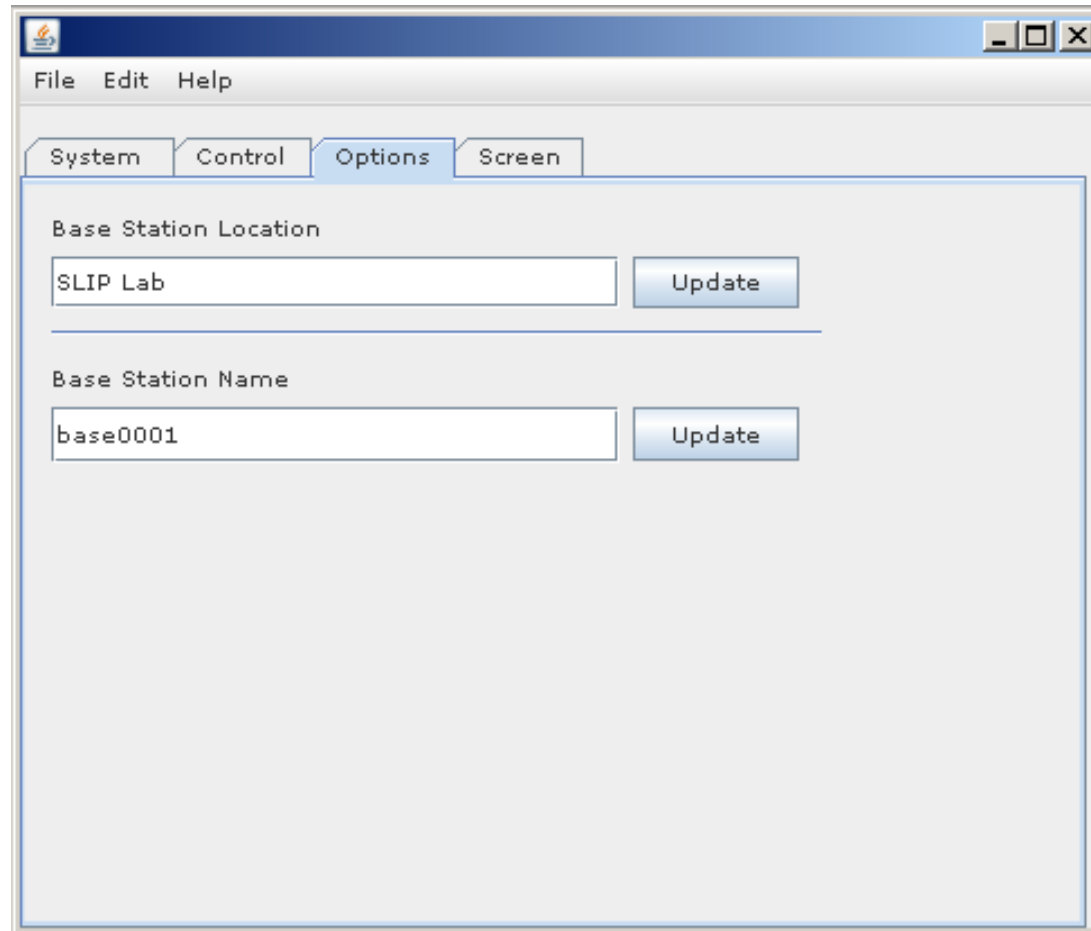


Port open Sent: 11 \ Received: 11

# Java Application : Control Panel



# Java Application: Options



The image shows a screenshot of a Java application window titled "Options". The window has a standard Windows-style title bar with minimize, maximize, and close buttons. Below the title bar is a menu bar with "File", "Edit", and "Help" options. The main content area features four tabs: "System", "Control", "Options" (which is currently selected), and "Screen".

Under the "Options" tab, there are two sections for editing base station information:

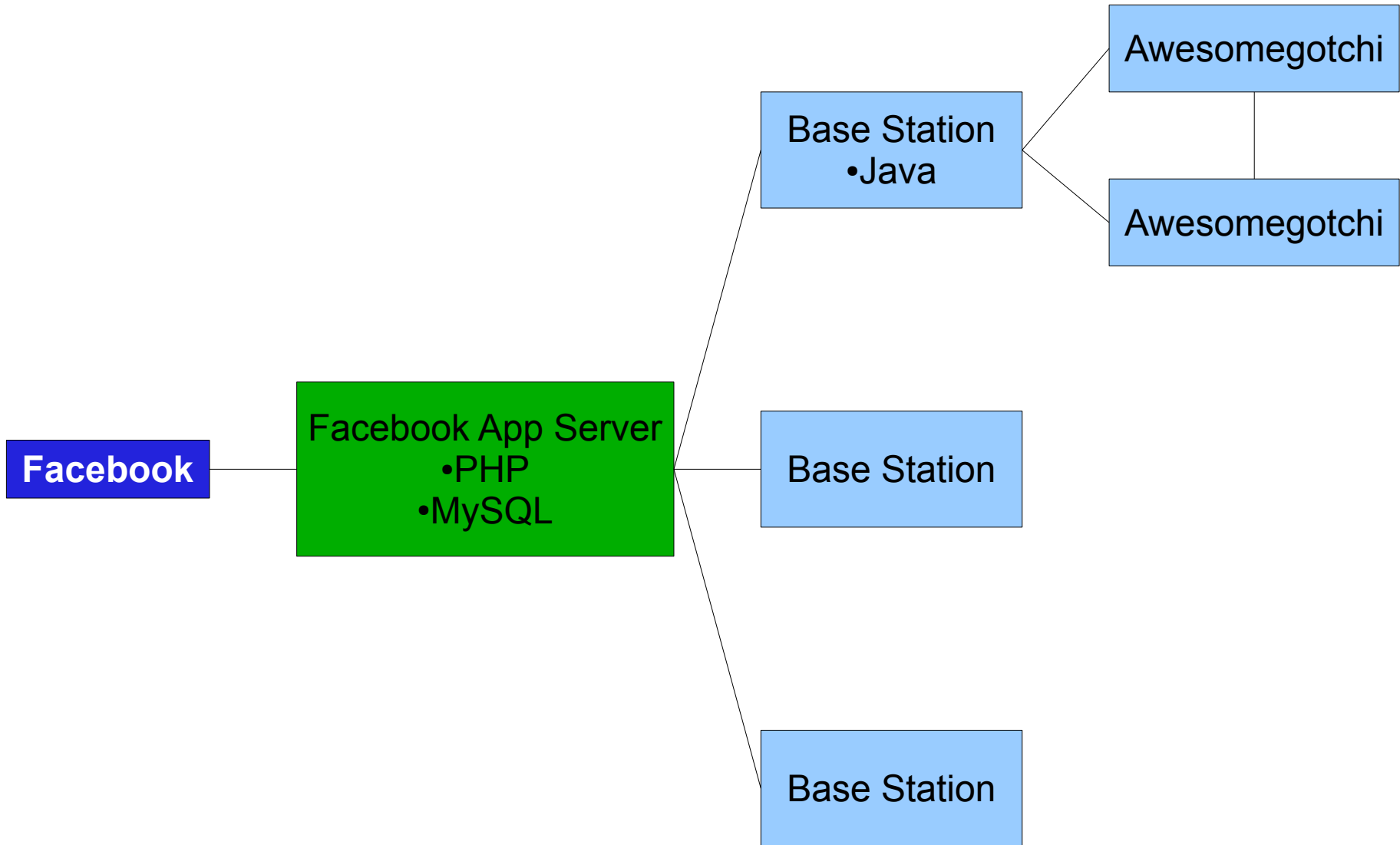
- Base Station Location:** A text input field containing "SLIP Lab" and an "Update" button to its right.
- Base Station Name:** A text input field containing "base0001" and an "Update" button to its right.

The application is running on a light-colored desktop background.

# Facebook Application

- Centralised database for all base stations and Prospeckz devices
- Communication relay between the base station and Facebook
- Managing Facebook users, their friendships, keeping record of the game results

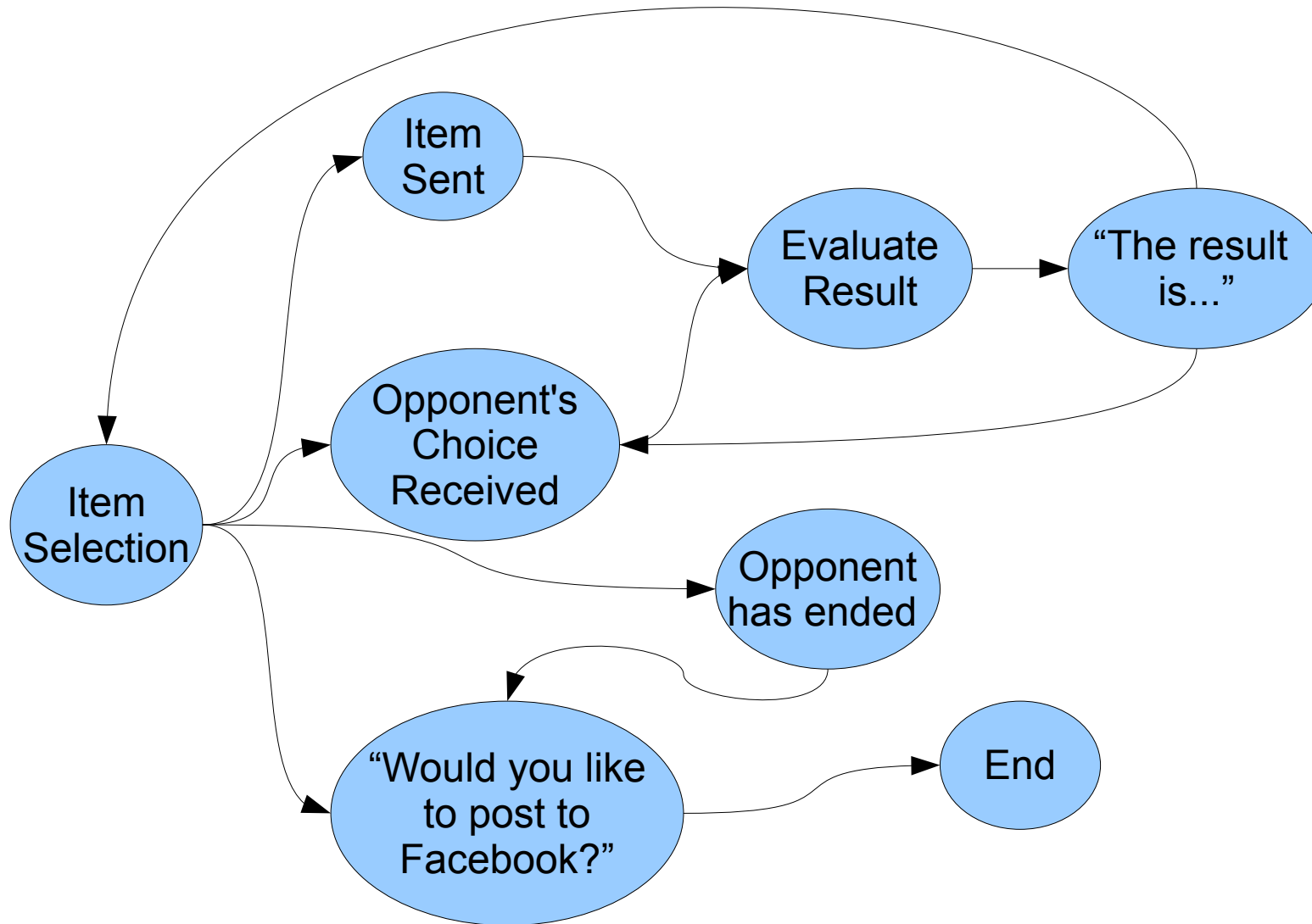
# Facebook Application



# Rock-Paper-Scissors Game

- A simple application running on the system we created for Prospeckz
- Compatible with the scheduler, menu, reliable transmission of packets
- Operates its own logic which is easily separable from other parts of the system
- Can be replaced with any other application

# Rock-Paper-Scissors Game



# Facebook and Prospeckz integration

