HomeOS

CSE 481m

April 4, 2011

Lots of tech in homes













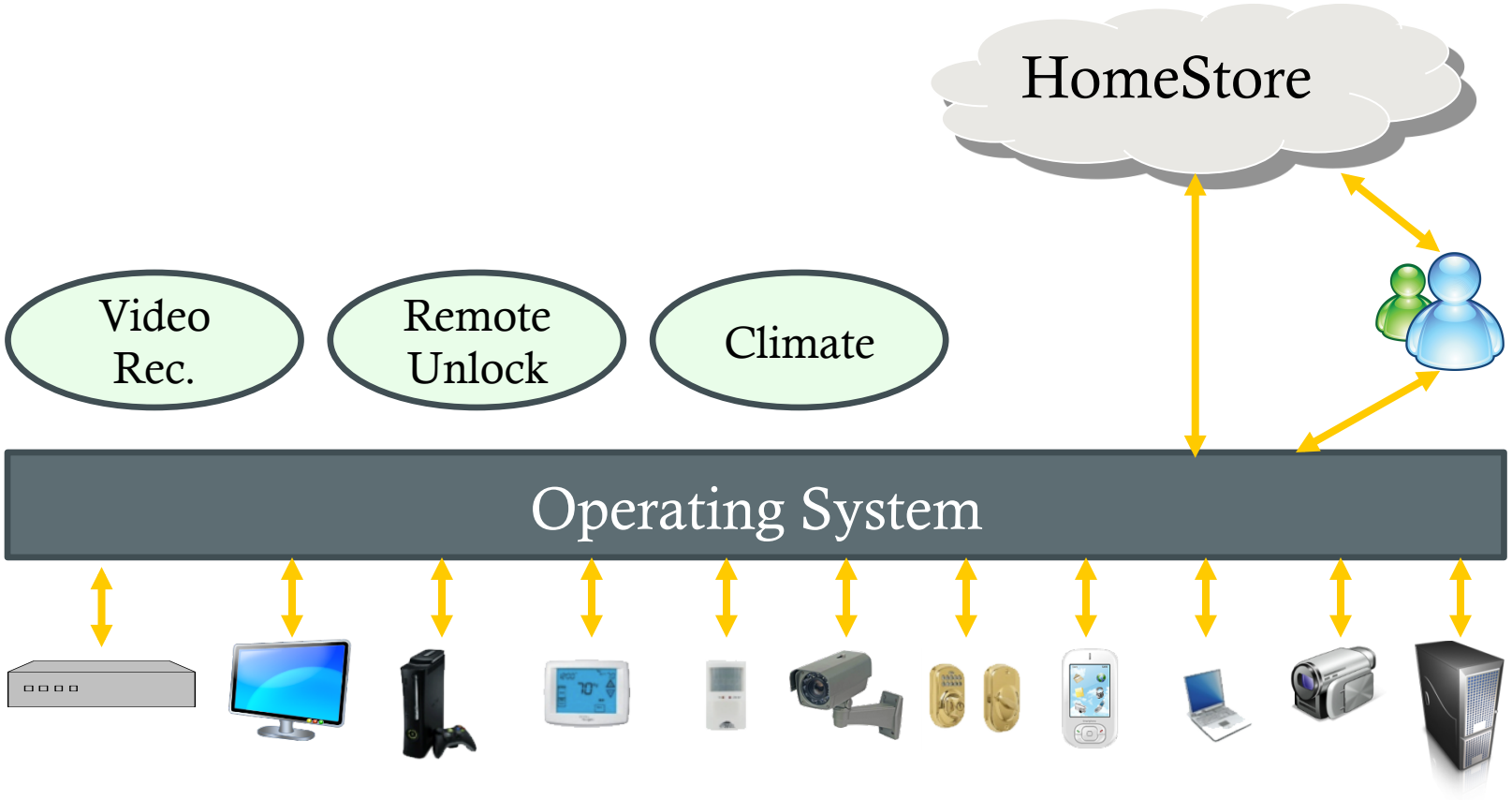

Problems with tech in homes

- You easily can't program it
- Why not?
  - Lack of standards
  - Diverse devices (most don't run code)
  - Different connectivity (ZigBee, Zwave, 802.11)
  - Sharing devices is hard
  - Users want different things

What HomeOS does

- Makes it easier to write apps for the home
  - Uses drivers to hide connectivity differences
  - Standardizes device interfaces
  - Standardizes user control of applications
  - Allows for constrained ‘sharing’ of devices

How it works


HomeStore
Video Rec.
Remote Unlock
Climate
Operating System

How it works

Application Logic

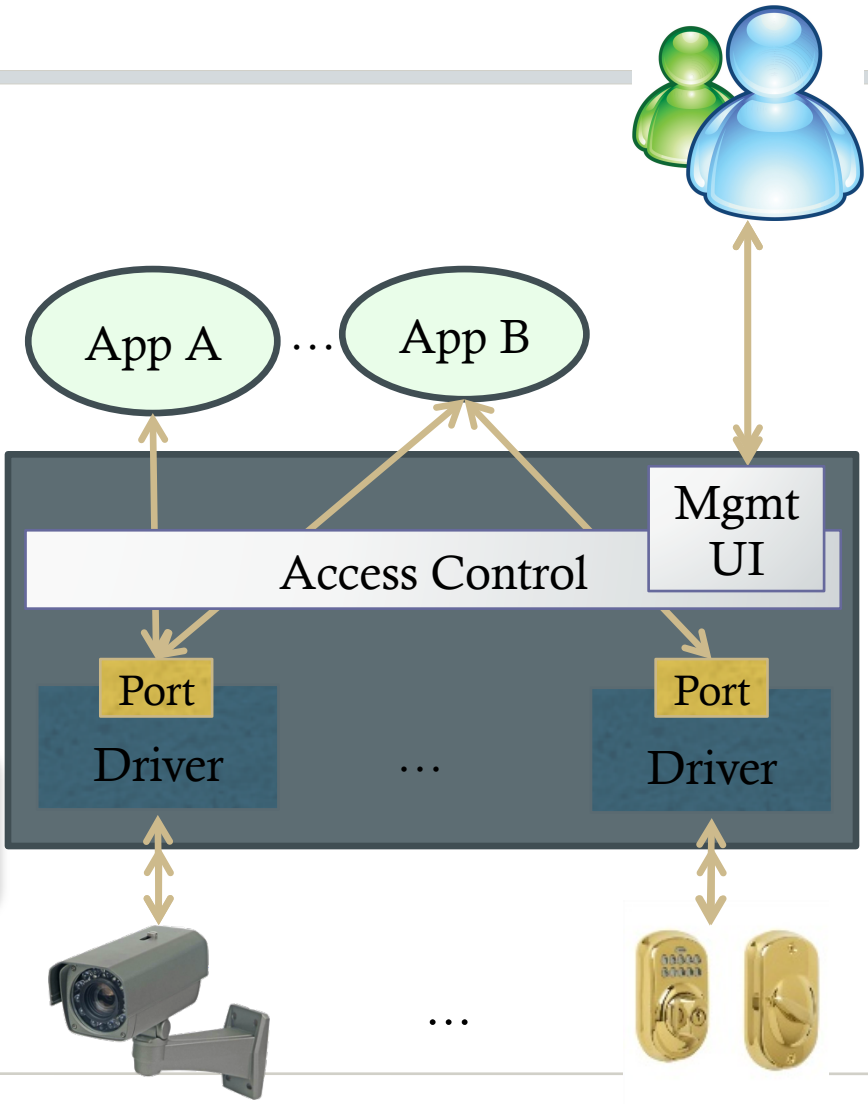
User Preference
Users' manage access control rules

Coordination
Access control mediates conflicts

Device
Standardize at functional layer

Topology
Logically centralize devices

Hardware
The actual devices in the house



What we gave you

- HomeOS
  - .NET project/library to make writing apps easy
  - Silverlight/WP7 SDK for GUI and phone development
- Includes
  - Drivers for: webcams, IP cameras, media server/player, z-wave devices, SMS notifications, face recognition, etc.
  - Sample applications

Applications

- Hopefully small pieces of code which orchestrate a series of devices
- Basic App Architecture
  1. Boot, set up any state (windows, connections, etc.)
  2. Look for required devices
  3. Once having found all devices, enter a loop
     a. See if relevant devices have showed up or left
     b. Do whatever useful thing it is supposed to do
  4. On quit, clean up state

Drivers

- Deal with connectivity and device specifics
  - Find the device and establish communication
  - Translate high level commands to low-level
- Export functionality as Roles & Ports
  - Advertise ports with relevant roles when devices are available
  - Remove ports when devices are no longer available

Finding Devices

- `PortRegistered()` and `GetAllPortsFromPlatform()`
  - Cycle through all ports on boot
  - Listen for new devices over time
  - Might hear about a new device more than once
  - Pick the ones you want
- Analogous `PortDeregistered()`
  - Listen for devices being removed

```
// ask for all the ports
IList<View.VPort> allPortsList = GetAllPortsFromPlatform();
foreach (View.VPort newPort in allPortsList)
    PortRegistered(newPort);
```

Ports & Roles

- Port
  - A handle to a device
- Role
  - Each port has one or more roles
  - Things like lightswitch, dimmerswitch, TV, media server, media player, etc.
  - What applications are actually written against

Invoking an Operation

```
private bool PlayContent()
{
    if( this.dmsPort == null ){
        logger.Log("{0} no DMS port found yet.", this.ToString());
        return false;
    }
    if( this.dmrPort == null ){
        logger.Log("{0} no DMR port found yet.", this.ToString());
        return false;
    }
    if (content.Count < 1)
    {
        logger.Log("{0} found no content on the DMS {1}.", this.ToString(), dmsPort.ToString());
        return false;
    }

    Uri media = content[0].Item2;

    //construct args
    List<View.VParamType> args = new List<View.VParamType>();
    args.Add(new ParamType(ParamType.SimpleType.text, "", media.ToString(), "uri"));

    //make call to port
    DmrPort.Invoke(RoleDmr.RoleName, RoleDmr.OpPlayName, args, this.ControlPort, DmrPortCapability, null);
    return true;
}
```