Mobile and Pervasive Computing
CNT5517 - Section 2D32 & CIS4930 - Section 2D78
Lecture 1

Syllabus Review & Planning

Dr. Sumi Helal

Computer & Information Science & Engineering Department

University of Florida, Gainesville, FL 32611

helal@cise.ufl.edu

Outline

- Welcome and personal introductions
- Understanding this class' range of skills
- Syllabus
- The term Project
  - Know each other: Why? Logistics?
  - What kind of term project?

Class Demographics

- Rolls indicate 68 students are registered with a few more being added. Stats below are based on attendance and not enrolment.

	Software	Hardware	Both
Undergraduate	16	1	3
Graduate Ph.D.	0	0	1
Graduate Masters	38	0	0

Know each other

Name	U/G	1st Skill	2nd Skill	Interest	AM/PM	Biz BG	Project Mangement	Present. Skills	Writing Skills
		Java - A	mCont. - M	Mobile Apps	AM	No	Little	Weak	Yes
		C - M		Mobile API's	PM	No	Yes	weak	No
				Sensor systems			No		
				Combined sensor/mobile					
				others					

How to know each other?

- Form a class social network (easy, quick but less structured).
- Utilize Google Drive (easy but must add all class emails to access list, very structured)
- Also utilize mailing list for the class in Sakai (being set up).

- Volunteers needed: No Volunteer, TA Miguel has already set up a Google drive Smart Folder.

Term Project?

- There must be a **problem** to which an adequate solution would make an impact.
- New problem, in the sense no one has ever thought of addressing it with mobile and/or pervasive technology.
- Known problem to which existing solutions are inadequate, or that your solution will compete with.
- There is a major market behind the problem which you can research and provide evidence for.

Mobile and Pervasive Computing

CNT5517 - Section 2D32 & CIS4930 - Section 2D78

Lecture 2

From the Newton to the iPhone

Introduction to Mobile Computing

Dr. Sumi Helal

Computer & Information Science & Engineering Department

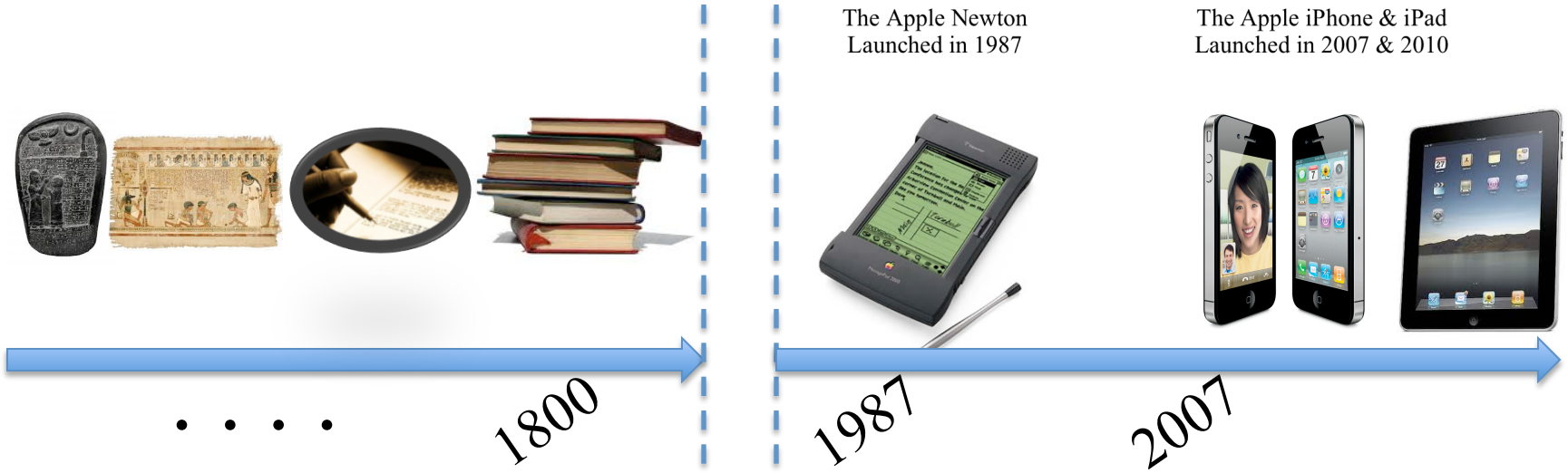
University of Florida, Gainesville, FL 32611

helal@cise.ufl.edu

Outline

- A short history of the PDA evolution
- First generation (1G) Mobile Platforms
- Limitations of 1G mobile platforms
- The emergence of 2G Mobile Platforms

From the Newton to the iPhone



- First platforms: Stones; papyrus scrolls & ink, then in 1800's the printed book
- Apple Newton as a PDA and a Tablet concept, redone two decades later as the iPhone and the iPad

The Mobile Platform Journey

The Newton

The beginning of the Journey

- Attempted to realize a Tablet concept by utilizing a **stylus** with a handwriting recognition as the main user interface.
- Stylus started a short period of “pen computing” developments and products that peaked and then faded into extinction, all within 20 years.
- The Tablet idea lived on however, and returned in the form of the iPad.

The Newton

The beginning of the Journey

- Newton OS consisted of:
  (1) a microkernel for what was then considered a low-power ARM RISC architecture,
  (2) operating system services written in C++ offering PDA manager services and handwriting recognition, and
  (3) a *NetwonScript* application development environment, which was an object oriented language.

Newton Successors: Motorola Marco & Envoy

Marco

Specs

- **Newton OS 1.3**
- 4MB ROM, 687KB Flash RAM
- 320x240 Monochrome LCD resistive touchscreen
- RS422 serial port
- Localtalk support
- 1 PCMCIA Slot (5V or 12V)
- 1 Sharp ASK infrared port
- 4 AA batteries, rechargeable NiCd batteries may be used
- First released January 1995
- It weighs 1.8 pounds and is 7.5 inches high, 5.8 inches wide and 1.4 inches deep
- Price: $900-1400

Similar to the Marco. Executive version.
Utilizes CDPD wireless data service (Cellular Digital Packet Data - a technology that offered packet data over circuit-switched networks).

Envoy
1996

The Newton, the Marco & the Envoy all crashed by 1997…

The Emergence of the Palm

- It is the *form factor* dummy!
- Must fit in your Palm, if it will have any charm
- Palm Inc. start up → US Roboics
  - The Palm Pilot PDA
  - The Palm OS
- Great Success, large volumes of sales, people started to actually use a PDA.

Microsoft Entering the race…

- Windows CE introduced around 1997 by Microsoft.
- Strong OEM support program encouraged many companies to adopt Windows CE.
- Support for pen computing and tablets.
- Marked the beginning of PDA bonanza: most products (labeled *hand-held*) used Windows CE, but each differed in the form factor.

1997-2002: A Zoo of Handhelds, Palms and Tablets

In the mean time:

The Advent of the Communicator

First Nokia Communicator 9000
(1996, GeOS, 397g)

Fifth & Last Nokia Communicator
(2007, Symbian OS, 210g)

First and Last Nokia Communicator

- Nokia introduced first communicator in 1996.
- Significant impact: *brought PDA concepts into cellular telephony.*
- PDA features + Telnet + phone + Fax

1G Mobile Platforms: Palm OS

- Single tasking environment with strong support for handwriting recognition, and extensive personal information management (PIM) support. Applications can be developed in C or C++.
- In 2005, Palm abandoned it OS and embraced Windows Mobile.

1G Mobile Platforms: Palm OS (continued)

- In another surprise move in 2009, Palm announced abandonment of Windows Mobile to start its new platform WebOS:
  - Under WebOS, all applications can be developed using HTML, Cascading Style Sheets (CSS) and JavaScript.
  - In 2010: HP acquired Palm, announcing plans to launch the TouchPad
  - For unclear/unknown reasons, HP killed the TouchPad in the same year it was launched!

1G Mobile Platforms: Symbian OS

- Evolved from Psion's EPOC, which was created in the mid 1990s. It consists of:
  - Symbian OS core and three UI frameworks to choose from: S60, UIQ and MOAPS.
  - MOAPS was developed and used by NTT DoCoMo, while the S60 and UIQ were licensed to a few companies including LG, Motorola, Nokia, Samsung and Sony Ericsson.
- In 2008, Nokia acquired Symbian, announced plans to form the Symbian Foundation to provide royalty-free open Symbian source based on the Symbian Core and the S60 UI.

1G Mobile Platforms: Symbian OS (continued)

- Symbian as a first generation smartphone OS achieved great success and dominated the market by a big margin for several years.
- In 2010, Gartner predicted that Symbian will continue to dominate the market in 2014.
- As it turned out, Android (a 2G platform) surpassed Symbian in 2011 although Symbian is still one of the top mobile OSes

1G Mobile Platforms: J2ME

- In 1999, Sun Microsystems' Java 2 Platform Micro Edition (J2ME) was introduced as an open standard and a lightweight virtual machine platform for mobile phones and other devices such as set-top boxes.
- J2ME was perceived as the next big development in mobile platforms, given the popularity and power of Java and the inherent promise of writing once, running everywhere.

1G Mobile Platforms: J2ME (continued)

- Now called: **Java Platform, Micro Edition**, or **Java ME**
- Java ME devices must be based on a device profile.
  - (Mobile Info Device Profile, or MIDP, for mobile phones, or Personal profile for set-top boxes)
  - Profiles are subsets of configurations:
    - CLDC: Connected Limited Device Configuration
    - CDC: Connected Device Configuration

1G Mobile Platform: BREW

- In 2001, Qualcomm introduced Binary Runtime Environment for Wireless (BREW)
- Competed with J2ME, to give CDMA devices an advantage through a faster and more capable/controlled platform.
- Not a virtual machine, not an OS, but a highly efficient firmware level code used by C or C++ applications.

1G Mobile Platforms in Retrospect

- J2ME was first serious attempt to create standardized computing and communication platform out of mobile phones.
- However, returns on investments out of J2ME (and other platforms) were hardly achieved.
- Consumer apps including games did not do very well, even though enterprise apps (company directory, simple CRM, field force automation, etc.) performed better in the US market.

Analyzing the limited Impact of 1G Mobile Platforms

- The idea of umbrella standard over OEM manufacturers did not work
  - E.g., J2ME was never a fully write once run everywhere (standard was difficult to implement)
  - Evolution of the hardware and the desire by OEMs to offer differentiators have fragmented the market and made portability a far goal to reach, at least in reality

Analyzing the limited Impact of 1G Mobile Platforms (cont'd)

- The mobile phone hardware was not ready
  - Lack of capabilities for power awareness, which limited the way J2ME and BREW were designed.
  - The hardware lacked innovative interaction devices. In fact, a QWERTY keyboard upgrade was big news for a handset, a move away from the tedious i^{th} element or T9 style keyboard input. This directly limited the UI of most applications developed on these platforms.

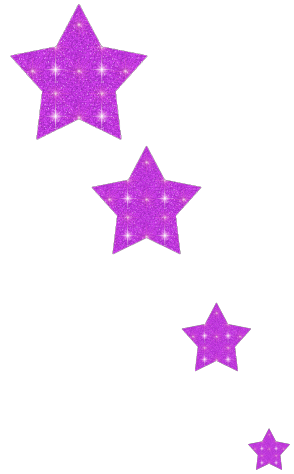
Analyzing the limited Impact of 1G Mobile Platforms (cont'd)

- The platforms were highly "regulated" and controlled by the OEM, the operator, or both
  - This left the developers communities with little room for innovations.
  - Some of these regulations were burdening and/or cost prohibitive to the developers, which made it difficult for them to bear any serious revenues out of their applications.
  - BREW, for instance, required costly testing of developed applications on each handset model a developer wished to support.

Analyzing the limited Impact of 1G Mobile Platforms (cont'd)

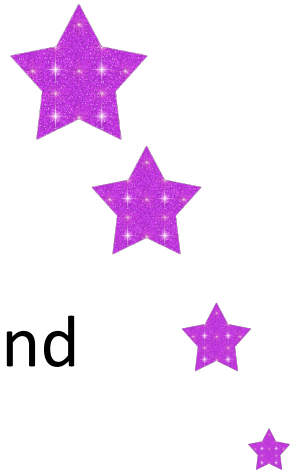
- Finally, the "*app ecosystem*" was missing.
  - Operators exaggerated in the scrutiny and selectiveness of the developers and their apps.
  - In fact, developers had to first earn the blessings of the operators before they could qualify to getting their apps on the "dock".
  - This posed a huge entry barrier to many developers, which hampered innovation.
  - Application listings were rudimentary to say the least, and the purchase process was very cumbersome to the consumer.

The Stars Aligned



1. Steve Jobs realized the elements of the missing piece: the ecosystem
2. Hardware Innovations
3. Emerging powerful platforms based on (1) and taking advantage of (2)

The Stars Aligned
The Hardware

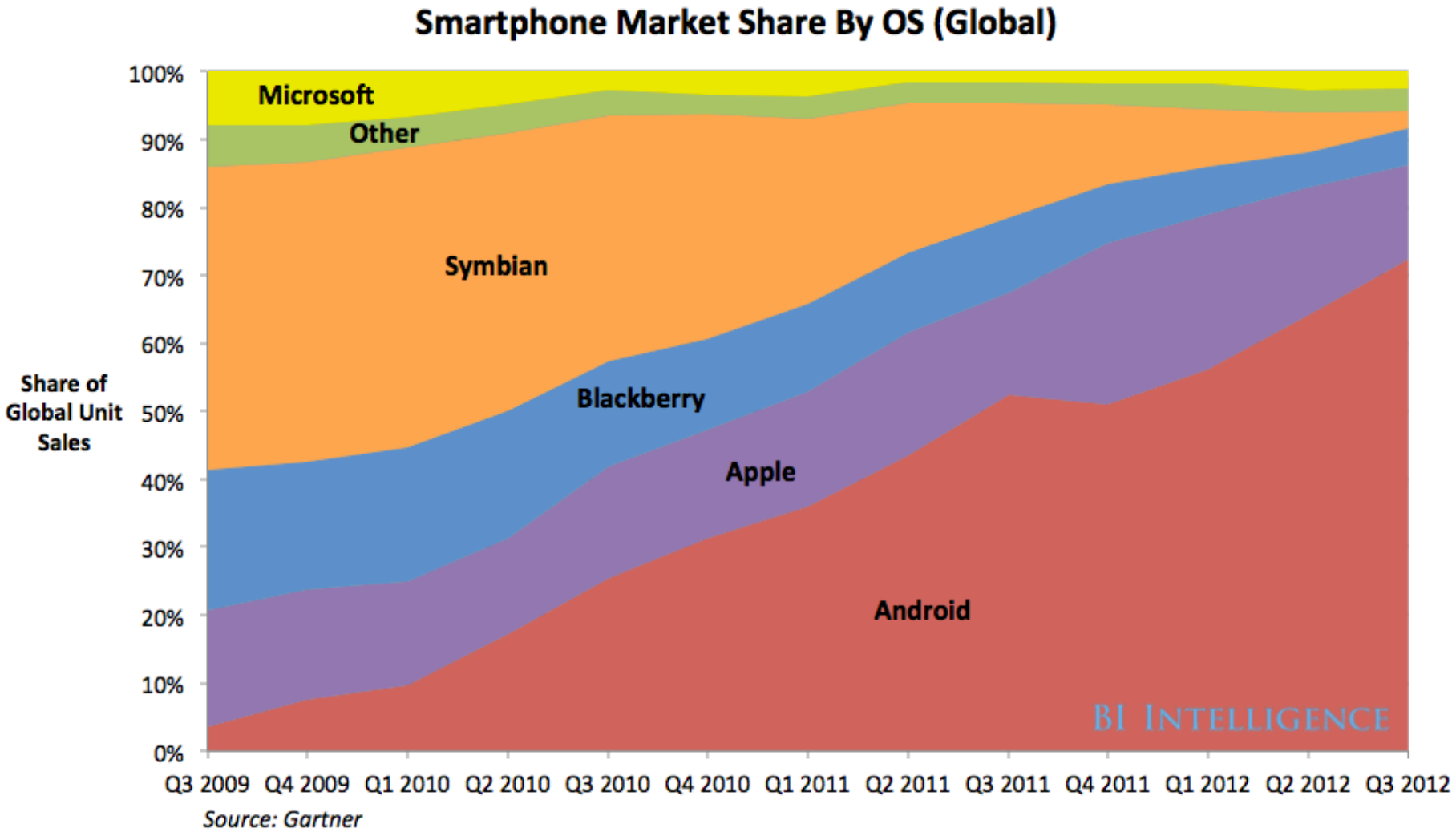


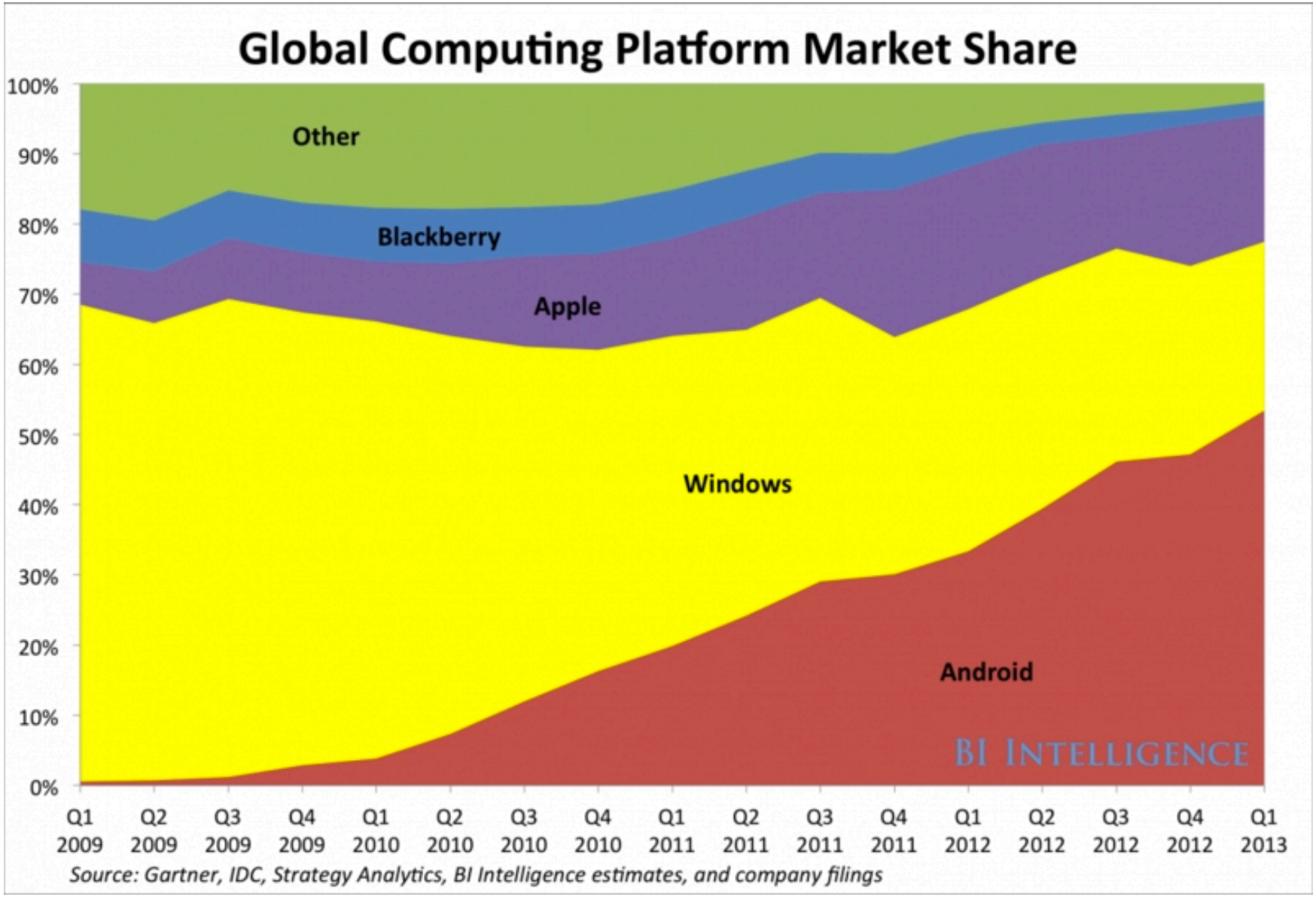
- Innovations in low-power mobile processors and higher quality displays
- Loosing (virtualizing) the keyboard/keypad
- Displays came out of age: the advent of the multi-touch display
  - Revolutionary interaction gestures reinvented the UI
- Low power, meter accuracy GPS and WiFi positioning technology enabled powerful LBS
- Embedded sensors (NFC, accelerometer, etc.) enabled powerful apps
- The Camera phone concept: high quality cameras

The arrival of 2G Mobile Platforms

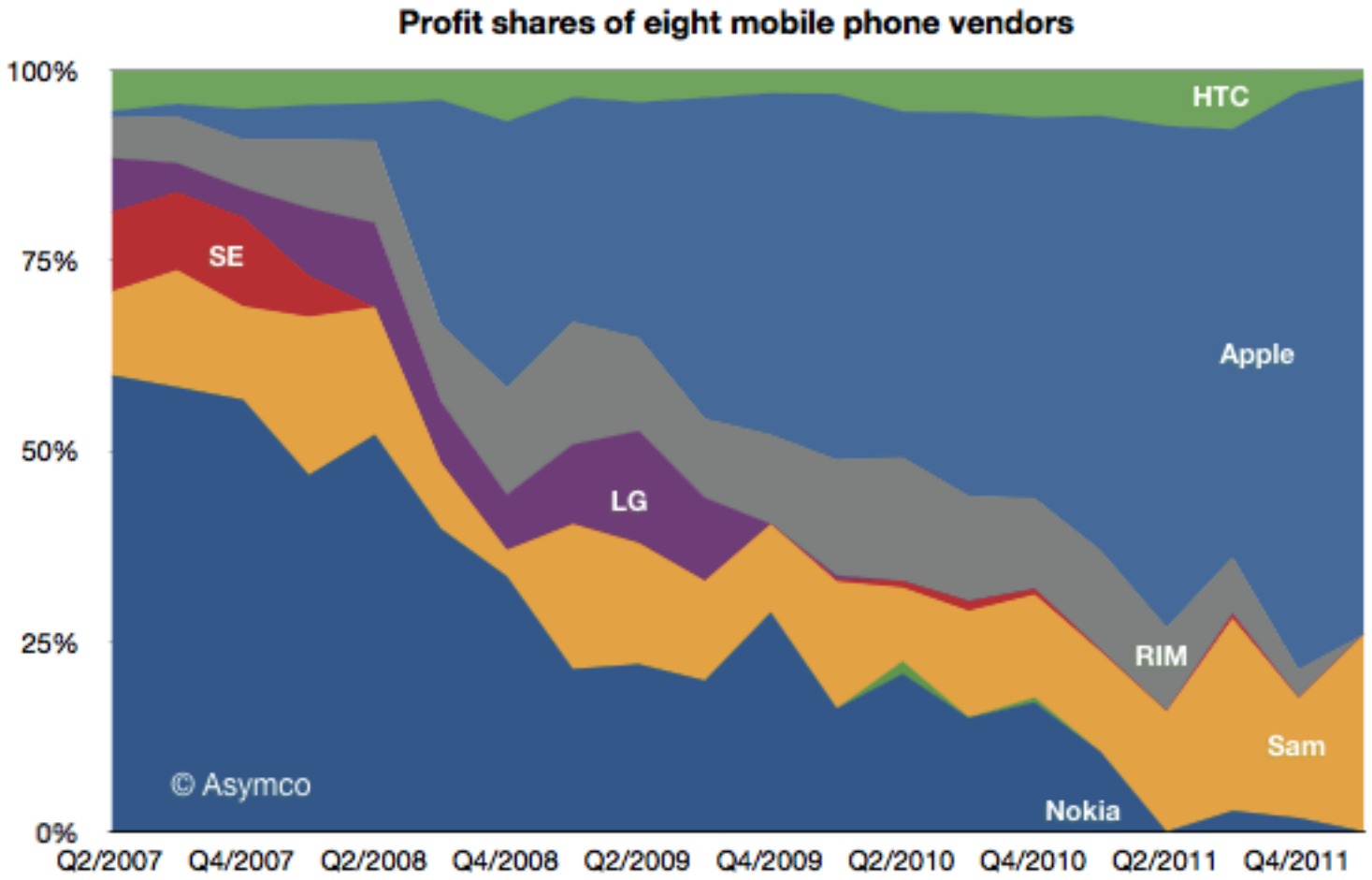
- As a consequence to (and along side) the hardware innovations and refinements, new software architectures and operating systems were created from the ground up to take advantage of, and to support the newest and greatest of mobile device hardware features.

Mobile Platform Market Share 2009-2012

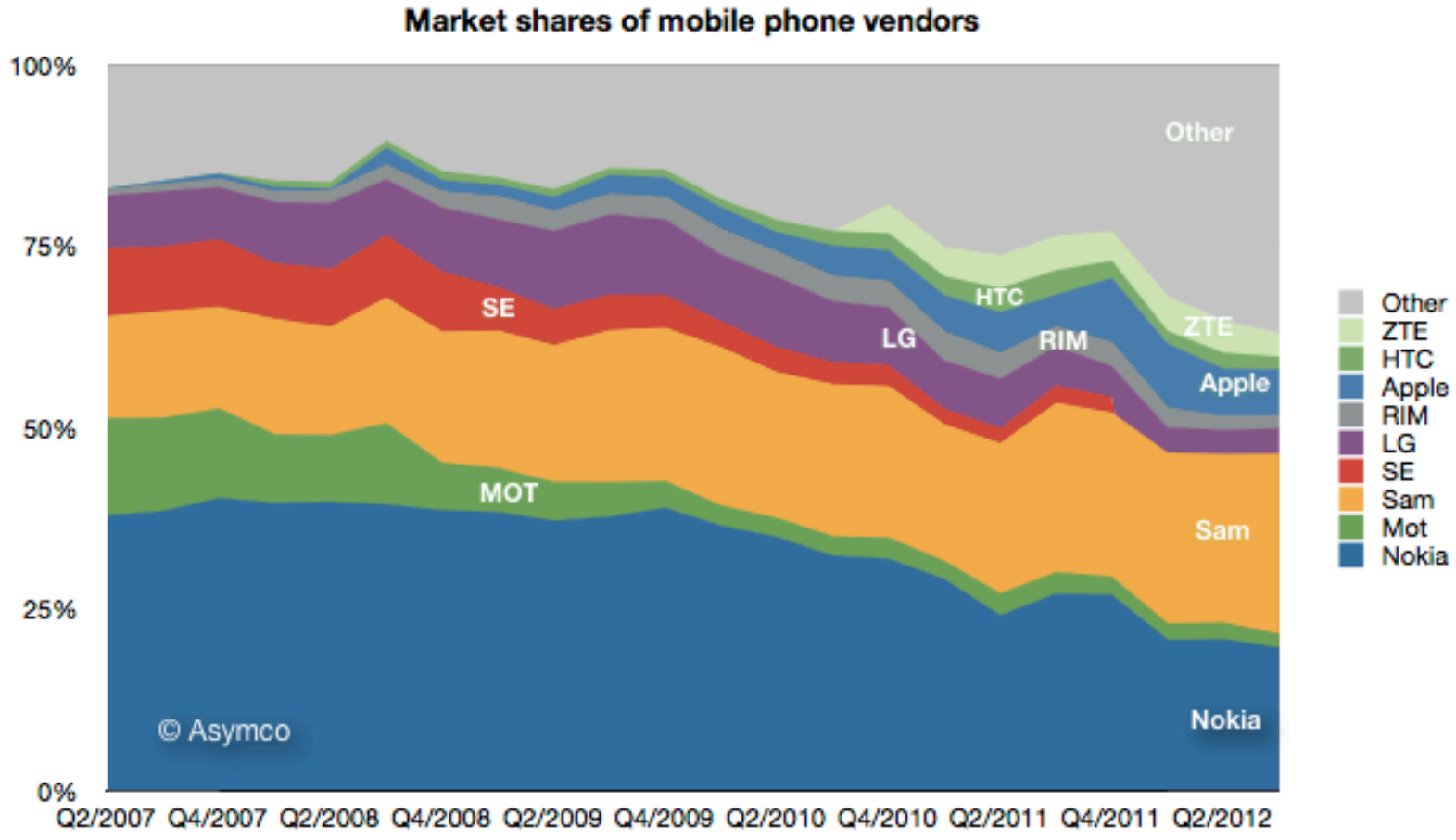



Global Computing Platform Market Share
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
Other
Blackberry
Apple
Windows
Android
BI INTELLIGENCE
Q1 2009
Q2 2009
Q3 2009
Q4 2009
Q1 2010
Q2 2010
Q3 2010
Q4 2010
Q1 2011
Q2 2011
Q3 2011
Q4 2011
Q1 2012
Q2 2012
Q3 2012
Q4 2012
Q1 2013
Source: Gartner, IDC, Strategy Analytics, BI Intelligence estimates, and company filings

Profit Share of Mobile Platforms



Market Share of Mobile Pone Vendors



Homework

- Readings - Mobile Platform book. Start with the intro but keep going.
- Research requirements for you to be able to publish and sell Android Apps. Same for iPhone. Summarize these requirements (app stores, platform experience, tools, contractual agreements, *Paypal*?, others).
- Find and lay your eyes on the simplest Hello World app in as many platform as you can.

Next Class

- Quick coverage of basics of Distributed Systems
- Limitation of the Mobile Environment
  - Devices, connections, interactions, others
- How to develop systems in mobile environment, or a Mobile Computing Models

Mobile and Pervasive Computing

CNT5517 - Section 2D32 & CIS4930 - Section 2D78

Lecture 3

Constraints & Requirements of Mobile Computing Systems

Introduction to Mobile Computing

Dr. Sumi Helal

Computer & Information Science & Engineering Department

University of Florida, Gainesville, FL 32611

helal@cise.ufl.edu

Lecture Overview

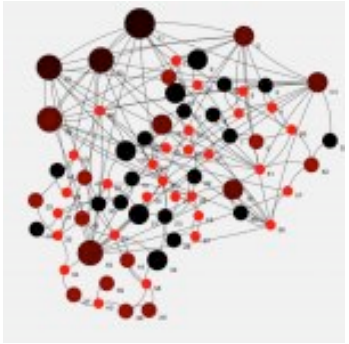
- Review of Distributed Systems

Distributed System Concerns

- **Reliability**
- **Availability**
- Dependability
- Security
- Privacy
- Responsiveness
- Utilization (and throughput)
- Scalability: vertical (concurrency) and horizontal (distribution)

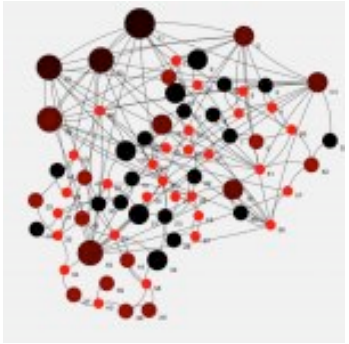
Fault-tolerance

Performance



Distributed System Concerns

- **Consistency**
- **Up-to-date-ness**
- Adaptability
- Re-configurability (SASO)
- Manageability
- Extensibility
- Transparency
- Portability
- more



Reliability

- Measure: $R(t) = Pr\{T > t\} = \int_t^{\infty} f(x)\,dx$
  - *t* is failure time
- Mean Time to Failure (MTTF)
- Extended Reliability = Availability
  - (1) Employing Redundancy (e.g., replication)
  - (2) Employing specific Recovery Procedures
  - In either (1) or (2):
    - First detect failure
    - Second Mask it via redundancy (swift), or Recover from it (noticeable downtime)

Availability

- Past:

$$A = \frac{E[\text{Uptime}]}{E[\text{Uptime}] + E[\text{Downtime}]}$$

$$X(t) = \begin{cases} 1, & \text{sys functions at time } t \\ 0, & \text{otherwise} \end{cases}$$

$$A(t) = \Pr[X(t) = 1] = E[X(t)].$$

$$A = \lim_{t \to \infty} A(t).$$

4 nine's = 0.9999

- Today:
  - Ubiquity enhances availability
    - Available through a variety of networks
    - Available through a variety of devices
    - Available through a variety of locations (e.g., while mobile?)
- Practical measure: The 9's Figure

Availability

Redundancy: Replication in Distributed Systems

- Consider data, how can we use replication of data to increase its availability and use?
  - Read One Write All
  - Read One Write All Available
  - Quorum Consensus
  - Others
- Caching
  - Cache coherency

Distributed Systems Intricacies

- **Synchronization**: multiple clocks (difficult to agree on exact time)
- **Concurrency**: multiple simultaneous accesses potentially conflicting.
- **Failures**: high probability of failures (too many components). Complex failure modes (single, multiple simultaneous, network partition, …)
- **Consensus**: difficult to reach consensus (odds includes failures, lack of synchronization, …)
- **more** …

Consistency
Transactions

- A transaction is a unit of program execution that accesses and possibly updates various data items. To preserve the integrity of data *ACID* must be ensured: Atomicity, Consistency, Isolation and Durability

T1		**T2**
1.	**read**(*A*)	
2.	*A* := *A* – 50	
3.	**write**(*A*)	
		read(A), read(B), print(A+B)
4.	**read**(*B*)	
5.	*B* := *B* + 50	
6.	**write**(*B*	

Consistency
Transaction ACID Properties

- **Atomicity** Either all or none of the transaction operations are completed.
- **Consistency** Execution of a (single) transaction preserves the integrity of the data.
- **Isolation** (*worry about the programmer*) Although multiple transactions may execute concurrently, each transaction must be unaware of other concurrently executing transactions. Intermediate transaction results must be hidden from other concurrently executed transactions.
  - That is, for every pair of transactions Ti and Tj, it appears to Ti that either Tj, finished execution before Ti started, or Tj started execution after Ti finished.
- **Durability**. After a transaction completes successfully, the changes it has made to the data persist, even if there are system failures.

Serializability

- **Schedule**: order of execution of transactions
- **Serial schedule**: No interleaved execution of transaction operations.
- **Serializable schedule**: a schedule that is equivalent in effect to a serial schedule.

Example Schedule 1

- Let T_1 transfer €50 from *A* to *B*, and T_2 transfer 10% of the balance from *A* to *B.*
- A serial schedule in which T_1 is followed by T_2 :

T_1	T_2
read(A)	
$A := A - 50$	
write (A)	
read(B)	
$B := B + 50$	
write(B)	
	read(A)
	$temp := A * 0.1$
	$A := A - temp$
	write(A)
	read(B)
	$B := B + temp$
	write(B)

Example Schedule 2

- A serial schedule where T_2 is followed by T_1

T_1	T_2
	read(A) $temp := A * 0.1$ $A := A - temp$ write(A) read(B) $B := B + temp$ write(B)
read(A) $A := A - 50$ write(A) read(B) $B := B + 50$ write(B)	

Example Schedule 3

- Let T_1 and T_2 be the transactions defined previously. The following schedule is not a serial schedule, but it is *equivalent* to Schedule 1.

T_1	T_2
read(A)	
$A := A - 50$	
write(A)	
	read(A)
	$temp := A * 0.1$
	$A := A - temp$
	write(A)
read(B)	
$B := B + 50$	
write(B)	
	read(B)
	$B := B + temp$
	write(B)

In Schedules 1, 2 and 3, the sum A + B is preserved.

Example Schedule 4

- The following concurrent schedule does not preserve the value of (*A* + *B*).

T_1	T_2
read(A)	
$A := A - 50$	
	read(A)
	$temp := A * 0.1$
	$A := A - temp$
	write(A)
	read(B)
write(A)	
read(B)	
$B := B + 50$	
write(B)	
	$B := B + temp$
	write(B)

Distributed Computing Models

- Client/Server Approach
- Multi-tier Approach
- Peer-to-Peer Approach
- Agent based systems
- Mobile Code (Applets, mobile agents, ..)
- Service-oriented computing
  - Service registration/discovery
  - Service composition
- Cloud Computing

Emerging Computing Models

- Cloud Computing
  - Cheaper to provide services to a thin client than to maintain a fat client in a changing world (service Science)
- Edge Computing (Cyber Foraging)
  - Besides the Cloud, all other computing infrastructures surrounding a client become simply resources to use

Limitations of the Mobile Environment

- Traditional limitations of the wireless network (90's to late 00)
  - frequent disconnections
  - limited communication bandwidth
  - heterogeneity of fragmented networks
- Limitations imposed by mobility
- Limitations of the mobile device

Limitation of the Wireless Network

Frequent Disconnections

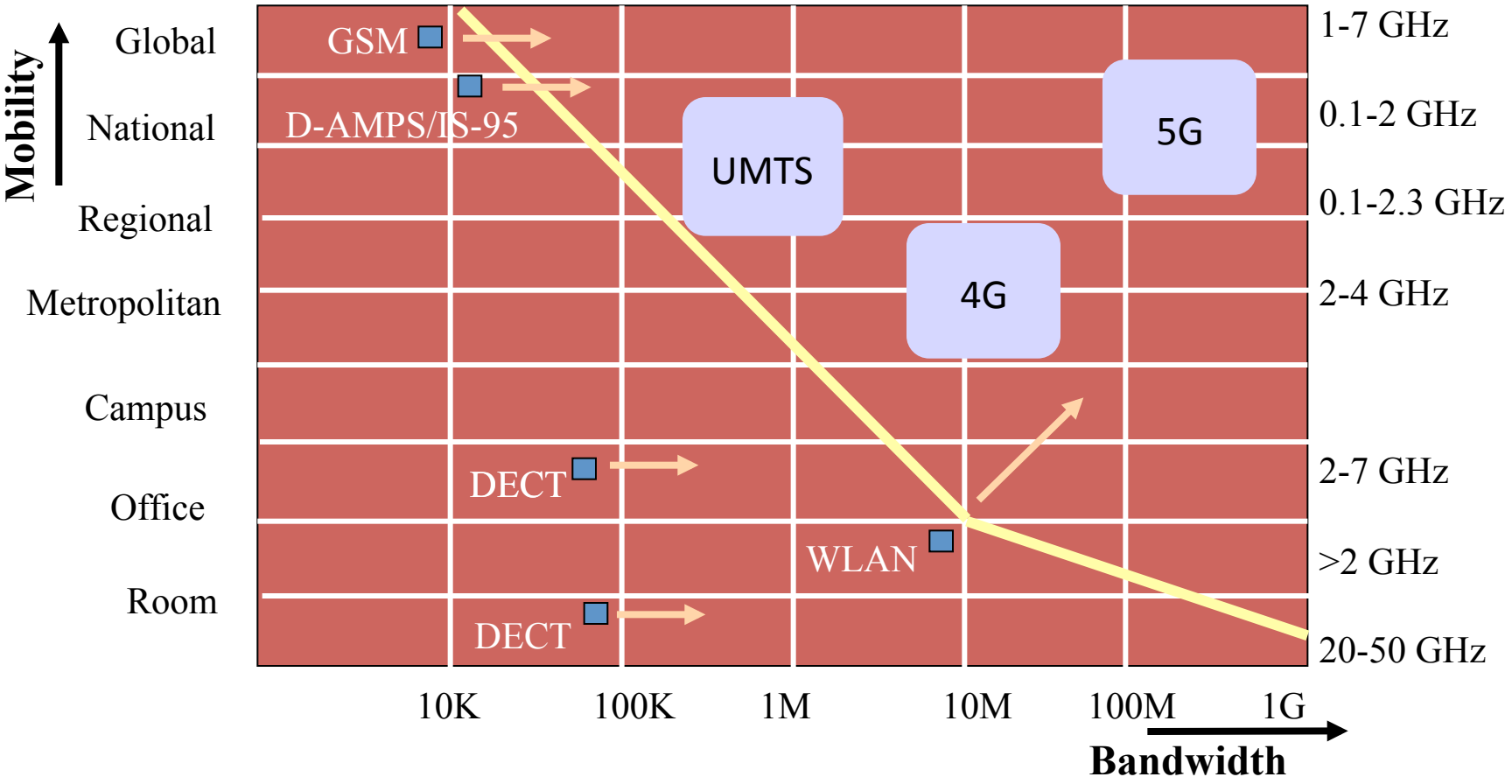
- Horizontal Handoff blank out (>1*ms* for most cellular)
- Vertical Handoff disconnection (or blank out (> 1000*ms* cellular to WLAN)
- Drained battery disconnections
- Roam-off disconnections
- Other disconnections:
  - Battery recharge down time
  - Voluntary disconnection (turned off to preserve battery power)
  - Theft and damage (hostile environment)

Limited Communication Bandwidth

- Orders of magnitude slower than fixed network
- Higher transmission bit error rates (BER)
- Uncontrolled cell population
  - Difficult to ensure Quality of Service (QoS)
  - Availability issues (admission control)
- Asymmetric bandwidth
- Limited communication bandwidth exacerbates the limitation of battery lifetime.

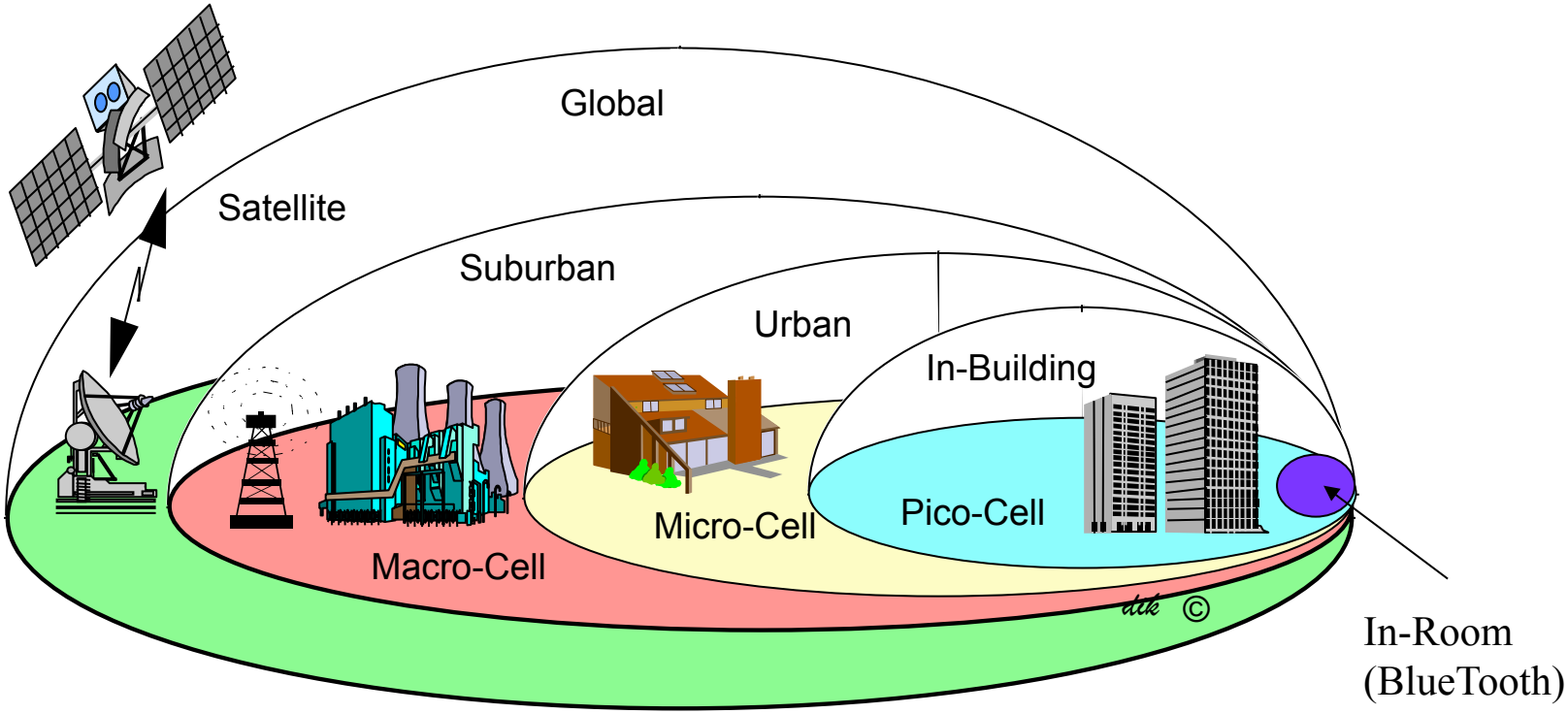
Wireless Network Convergence

Mobility-Bandwidth Trade-offs



Limitation of the Wireless Network

Heterogeneity of Fragmented Networks



Limitation of the Wireless Network

Heterogeneity of Fragmented Networks

- Support for vertical handoff is required
- Session maintenance over network overlays is complicated
- Difficult to manage security across heterogeneous networks.

Limitations of the Mobile Computer

- Short battery lifetime
- Subject to theft and destruction => unreliable
- Highly unavailable (normally powered-off to conserve battery)
- Limited capabilities (display size, memory, input devices)

Limitations Imposed by Mobility

- **Lack of mobility- and activity-awareness by applications**
  - inherently transparent programming model (object-, components-oriented, but not aspect-oriented)
  - lack of API & programmability support for context awareness
- **Lack of mobility-awareness by the system**
  - *network*: existing transport protocols are inefficient to use across heterogeneous mix of fixed/wireless networks
  - *Session*: inappropriate for the wireless environment and for mobility

Mobile and Pervasive Computing

CNT5517 - Section 2D32 & CIS4930 - Section 2D78

Lecture 4

Mobile Computing Models

Dr. Sumi Helal

Computer & Information Science & Engineering Department

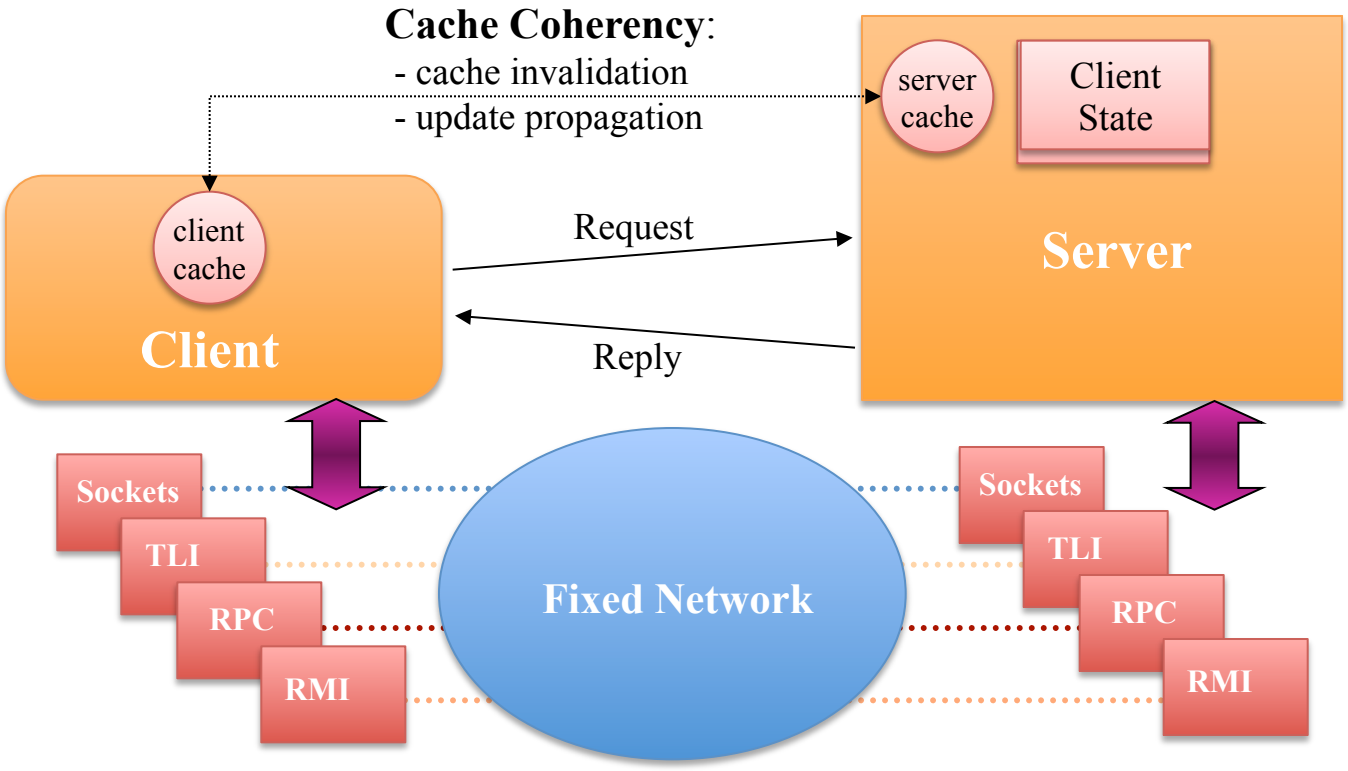
University of Florida, Gainesville, FL 32611

helal@cise.ufl.edu

Outline

- Client/Server Computing Model
- Mobile Computing Models
  - Unaware Client/Server Model
  - Client/Proxy/Server Model
  - Thin Client/Server Model
  - Disconnected Operation
  - Mobile Agents
  - Opportunistic Computing Model
  - Broadcast Desks
  - Cloud-Mobile & Cyber Foraging Models
- Cast Studies:
  - Coda
  - Odyssey

Review: Client/Server Computing



Client/Server Design

Review: Client/Server Computing

- Stateless/statefull client/server design
- Caching and cache invalidation
  - server invalidates client cache *and/or*
  - client requests server to validate its cache.
  - file system caching: writes => update propagation
- Connectionless/Connection-oriented design
- TCP/IP & Interfaces
- Other issues: multi-threading &deadlocks

Fixed Network Client/Server Assumptions

Review: Client/Server Computing

- Client Connectivity
  - client is always connected with availability comparable to the server's. Server can always invalidate the client cache
- Server Availability & Reliability
  - server is highly available. Reliable if stateless (but state info is exchanged in every C/S interaction), or if implements recovery procedures (may require client availability)
- Network
  - fast*, reliable*, BER < 10^{-6}, bounded delay variance

Taxonomy of Client/Server Adaptations

- **System-transparent, application-transparent**
  - The conventional, "*unaware*" client/server model
- **System-aware, application-transparent**
  - the client/proxy/server model
  - the disconnected operation model
- **System-transparent, application-aware**
  - dynamic client/server model
  - the mobile agent (object) model
- **System-aware, application-aware**

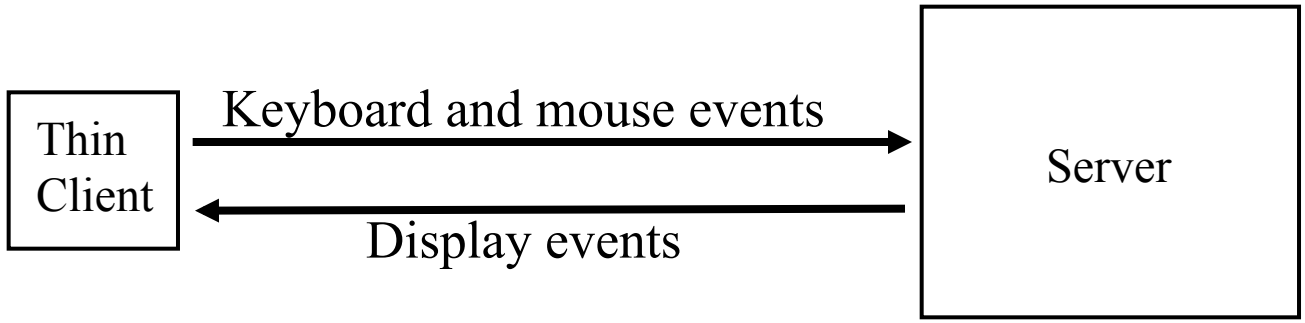
The *Unaware* Client/Server Model

- Full client on mobile host and full server on fixed network (SLIP/PPP C/S)
- Client and Server are not mobility-aware
- Client caching does not work as the client can be disconnected when the server invalidates the cache
- Not reliable and of unpredictable performance
- *Requires special cache invalidation algorithms to enable caching despite long client disconnections*

The Client/Proxy/Server Model

- Adding mobility-awareness between the client and the server. Client and server are not mobility-aware.
- Proxy functions as a client to the fixed network server, and as a mobility-aware server to the mobile client
- Proxy may be placed in the mobile host (Coda's Venus), or the fixed network, or both (WebExpress)
- Application-dependent
- Proxy embodies knowledge of limitations of the client and the wireless network and the transformations and adaptations necessary to achieve good performance.

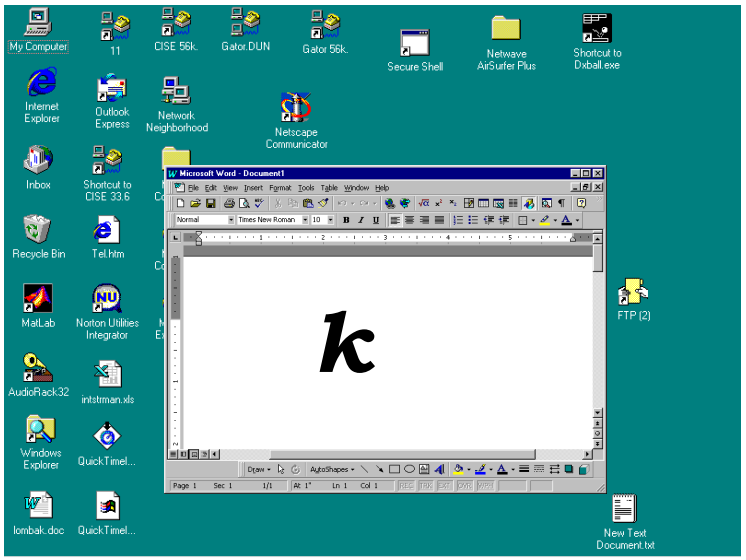
Thin Client/Server Model



- Thin client fits in resource poor mobile devices
- Requires at least weak connection, but generates bounded communication traffic
- Examples: CITRIX WinFrame and ICA thin client; InfoPad

How does T/C Work?

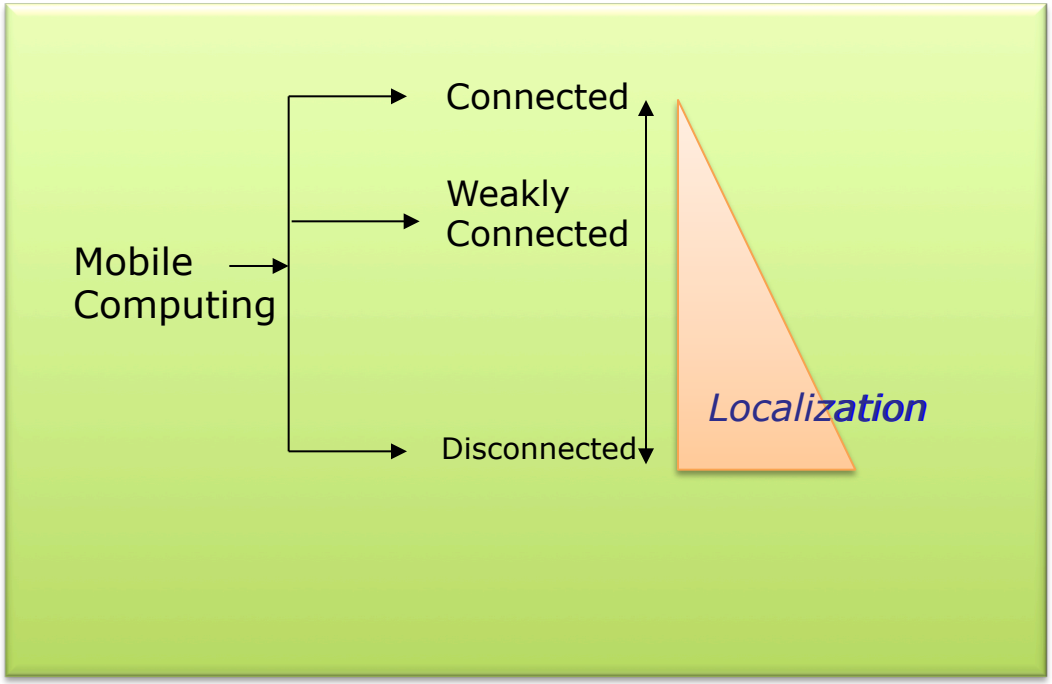
1. Type k
2. Send Keystroke “k” to word
3. Grab the window as a bitmap
4. Take the difference btw. bitmaps
5. Compress & Send difference to client
6. Decompress difference
7. Overlay the difference



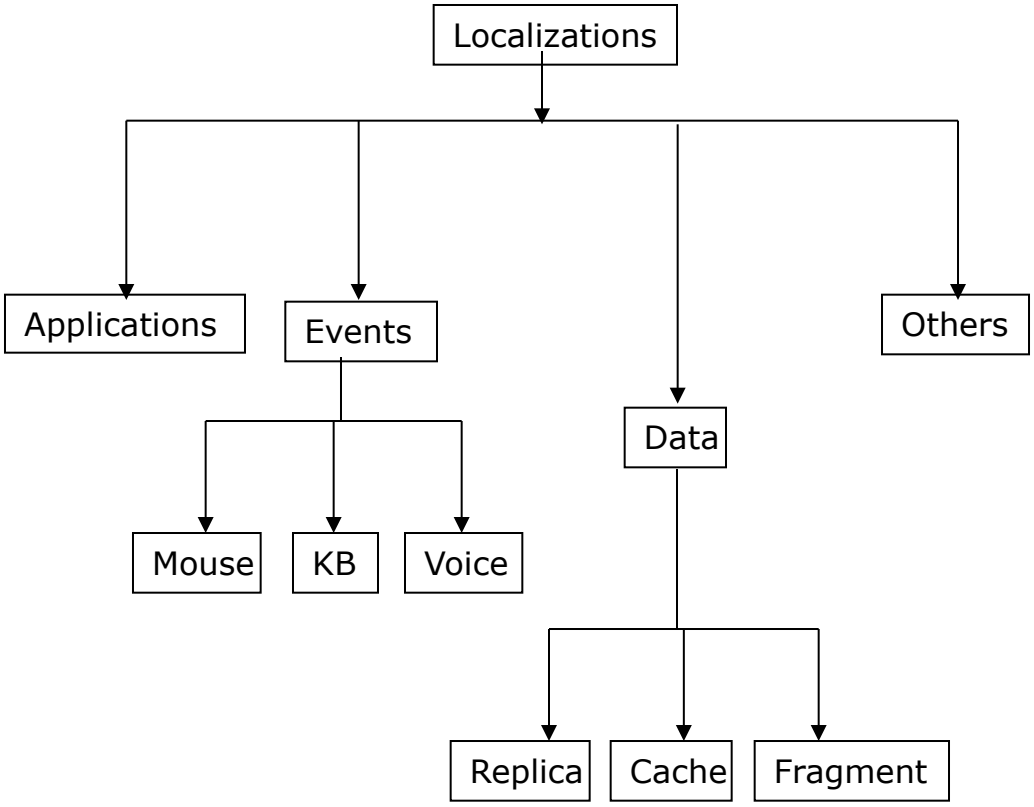
Wireless T/C Challenges

- Active Components
- Intense Interactions
- Total Disconnection

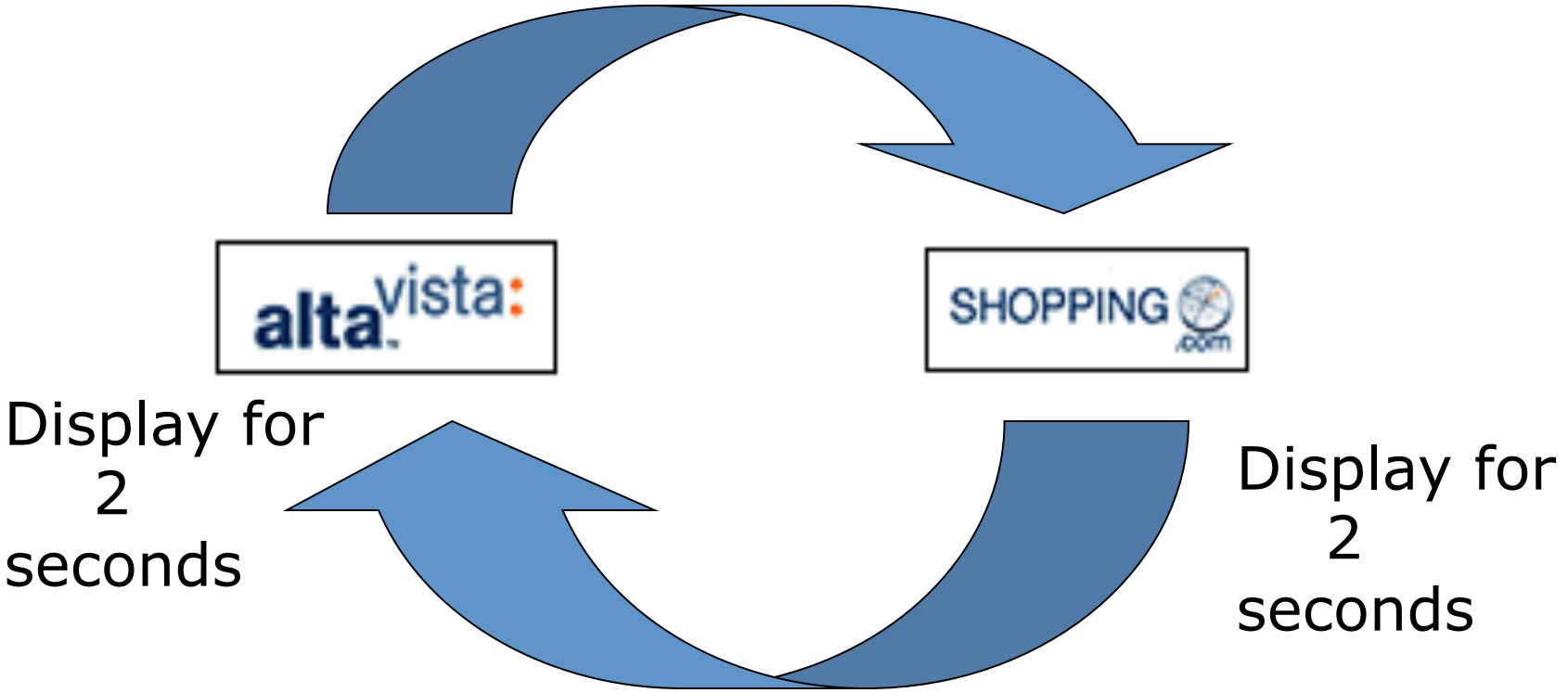
T/C Localization Gradient



Localization Space



Active Components

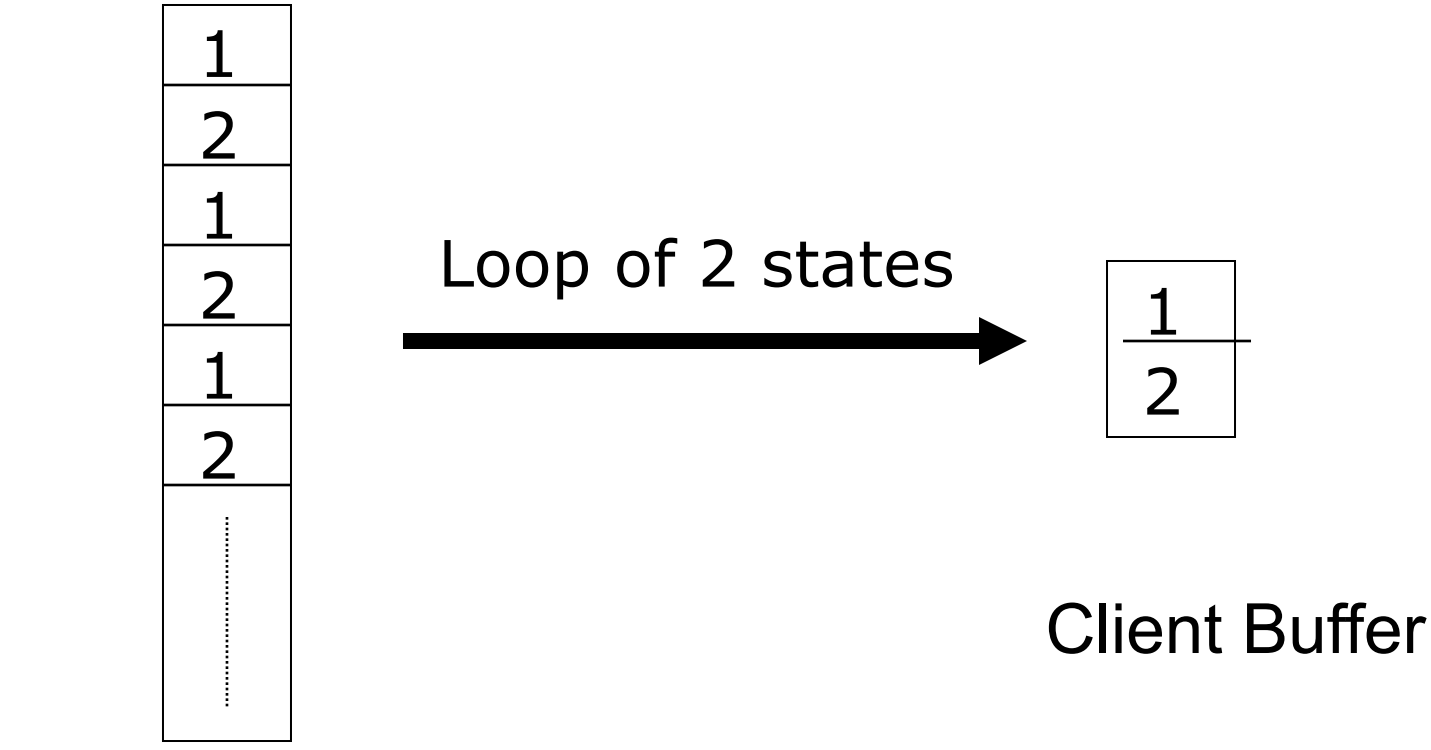


General Approach

- Detect recurring displays
- Extract active components separately
- Client simulates active components locally
- Transfer only non-repeating parts of the screen

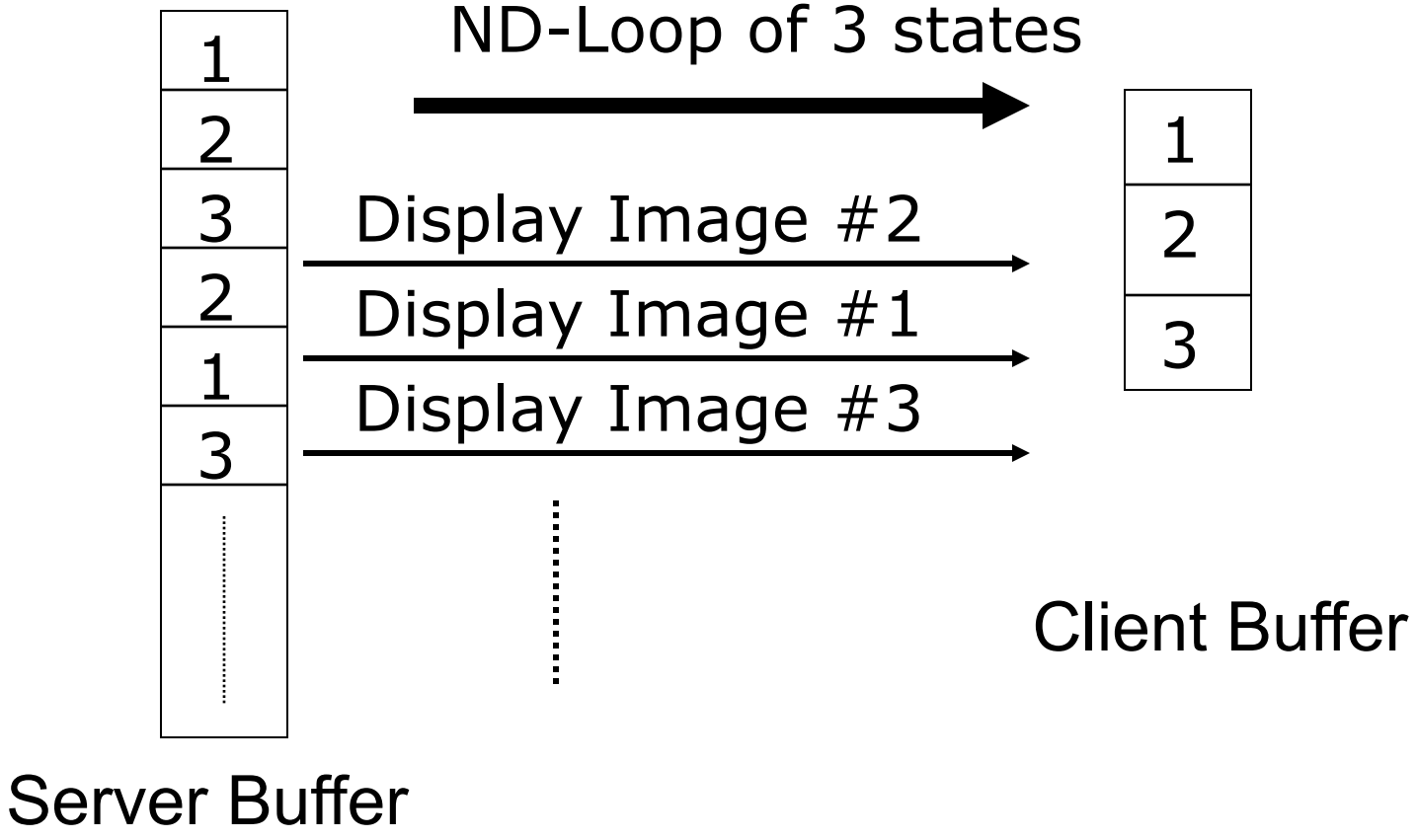
Loop Detection

Deterministic Loops



Loop Detection

Non-Deterministic Loops



State Explosion

- More than two active components
  - Different number of states
  - Different frequencies
  - Two 2-state active components cannot be captured in four states
  - With 3 active components, a buffer of >100 images required

Scan Line Algorithm
Aksoy et al*

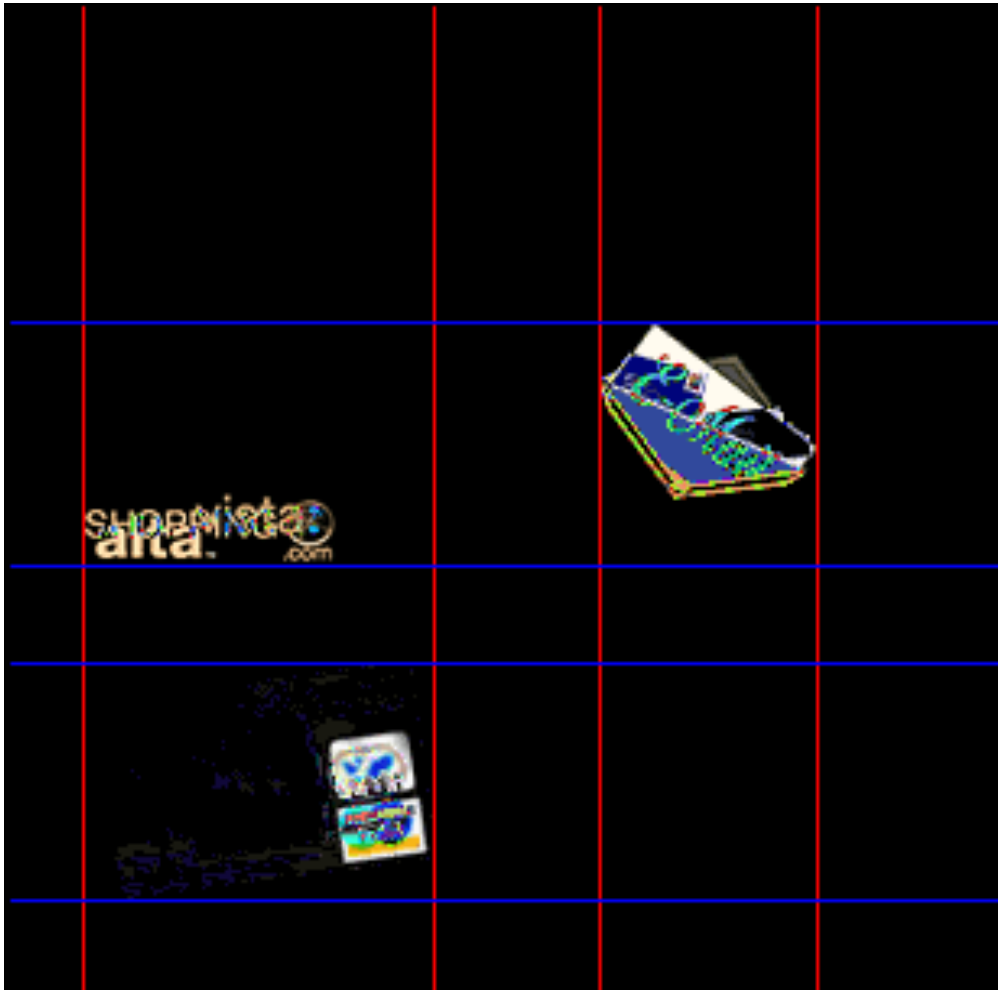
- Detect active components on each buffered image
- Pass scan lines -> Enclosing rectangles
- Vertical and horizontal scan lines can produce extra active components or big enclosing rectangles

C. Aksoy and A. Helal, "Optimizing Thin Clients for Wireless Active-media Applications," Proceedings of the Third IEEE Workshop on Mobile Computing Systems and Applications (WMCSA'00), held in Monterey, CA, Dec 7-8, 2000. (pdf)

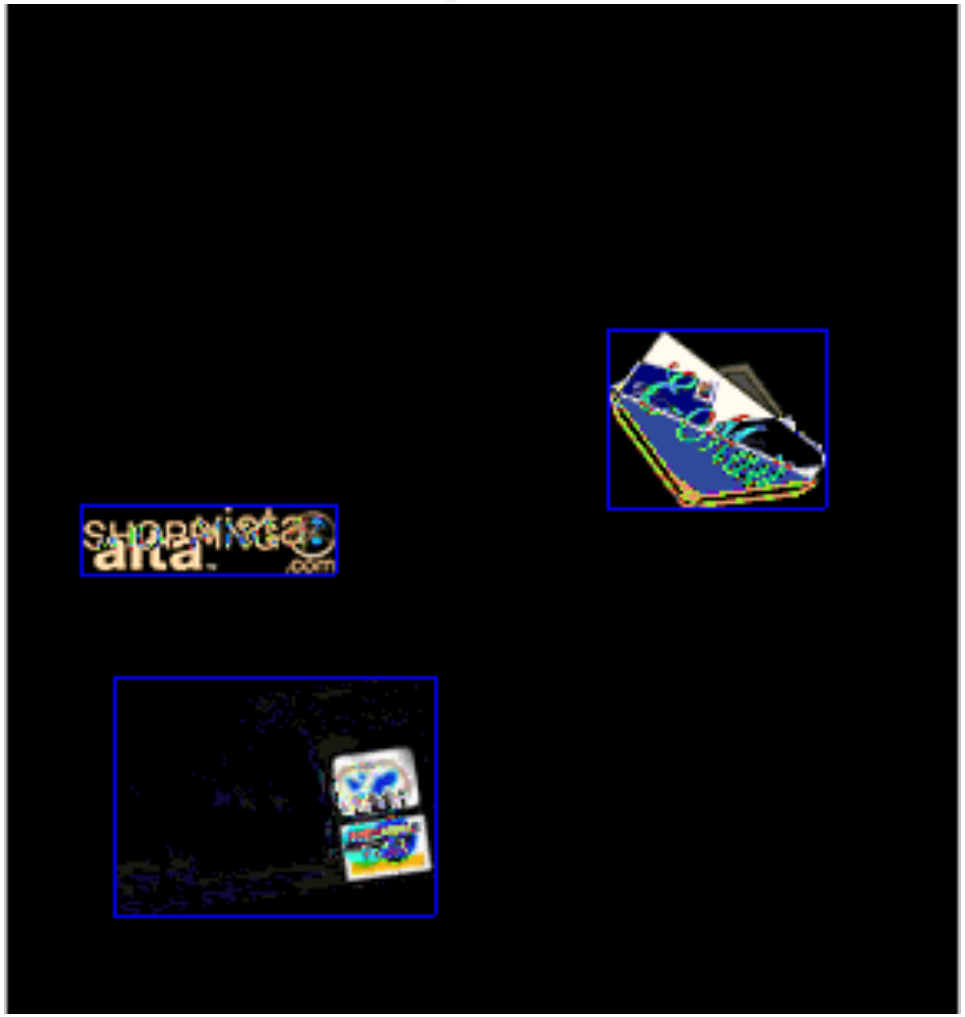
Scan Lines


TEST-2 HTML - Microsoft Internet Ex...
Links
Address
t\TestPages\5.html
Go
File
altavista:
Sony Card
VISA
Done
Local intranet

Scan Lines



Second Step Scan Lines



AC Extraction

- For each active component
  - Search buffer for the same enclosing rectangles
  - Compare the bitmaps in the rectangles to detect states
  - Use buffer timing information to extract timing of the active component

Active Components

- 32,129,29,101
  - States 1 and 2
  - Timing
    - State 1 every 2513 ms.
    - State 2 every 2494 ms.
- 187,232,142,182
  - States 1
  - Timing
    - State every 3613 ms.

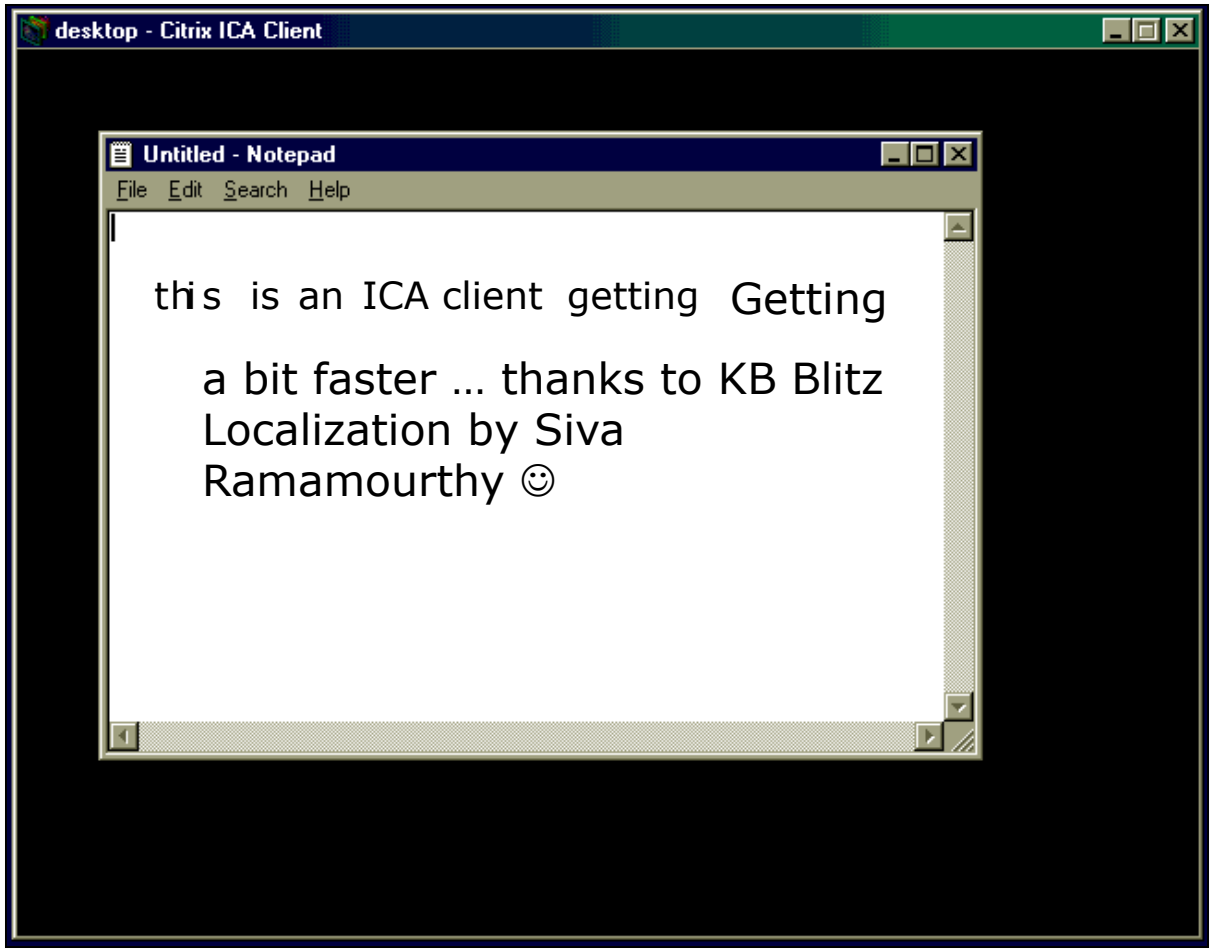
Keyboard Localization

- Localization begins only if intense KB activity is detected (KB Blitz, Ramamourthy et al)
- Then KB is fully localized and periodically synced and re-assessed
- The KB Blitz test involves
  - choosing KB-Blitz Window (how much of typing needs monitoring)
  - the kind of keys typed, and
  - the speed of the typing

KB-Blitz Localization Requirements

- The localization system should implement as much of its functionality at the server
- Display of localized typing must resemble the corresponding display when typing is handled by server
- Subtle features should be provided to create user awareness of any localization taking place

Keyboard & Mouse Localization

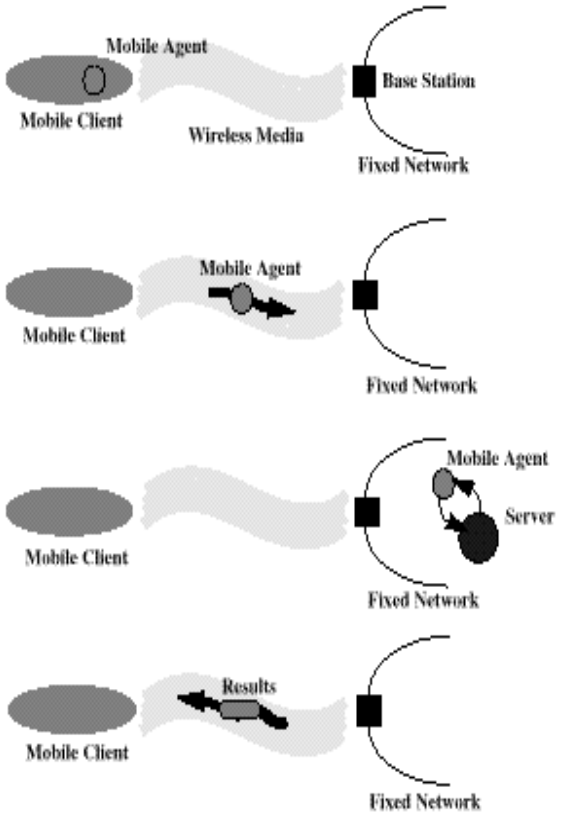


The Disconnected Operation Model

- **Approach I:**
  - Provide full client and a thin version of the server on the mobile platform. In addition, needed data is replicated into the mobile platform. Upon reconnection, updated replicas are synchronized with the home server. Conflict resolution strategies are needed (**Coda/Venus & Oracle Lite**)
- **Approach II:**
  - Provide a full client and a mobility agent that intercepts requests to the unreachable server, emulates the server, buffers the requests, and transmit them upon reconnection (**Oracle Mobile Agents**)

The Mobile Agent Model

- Mobile agent programmed with platform limitations and user profile receives a request; moves into the fixed network near the requested service
- Mobile agent acts as a client to the server, or invokes a client to the server
- Based on the nature of the results, *experienced* communication delays, and programmed knowledge, the mobile agent performs transformations and filtering.
- Mobile agent returns back to mobile platform, when the client is connected.



Opportunistic Computing Model

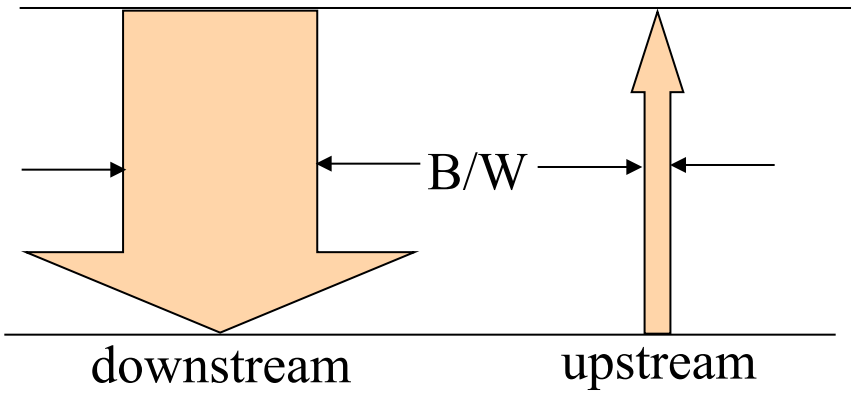
- An ad-hoc network of mobile devices is formed, each device offers services
- Some services may be of interest to mobile users, or other services
- Impromptu service composition based on opportunity rules and user interactions
- Service decomposition and tear down

C. Lee, A. Helal, N. Desai, V. Verma, and B. Arslan, "Konark: A System and Protocols for Device Independent, Peer-to-Peer Discovery and Delivery of Mobile Services", the IEEE Transaction on Systems, Man and Cybernatics, Volume 33, Number 6, November 2003, pp682-696.

Broadcast Disks

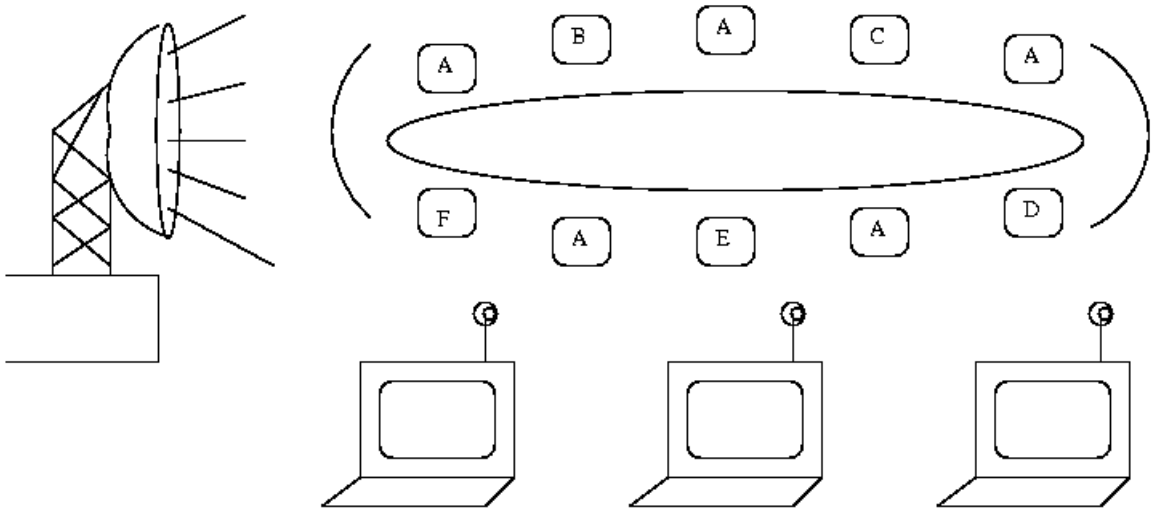
- Push data dissemination virtual device
- Indexing on air
- Client caching strategies and cache invalidation algorithms

Push/Pull Data Dissemination



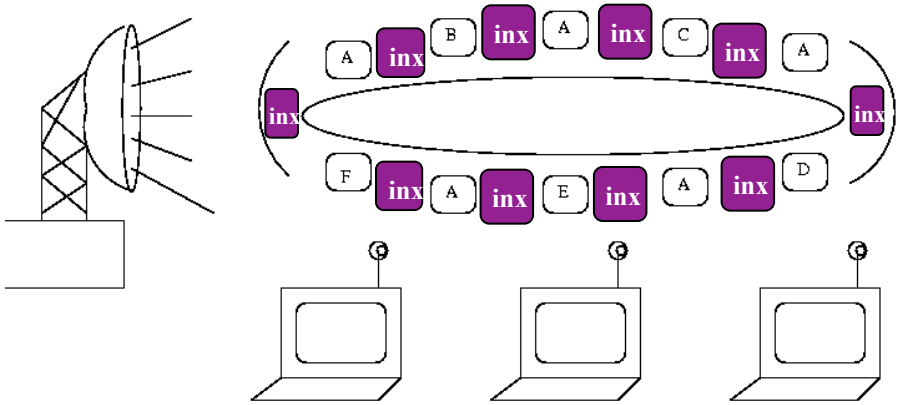
- Pull data delivery: clients request (validate) data by sending uplink msgs to server
- Push data delivery: servers push data (and validation reports) through a broadcast channel,to a community of clients

Broadcast Disks



Periodic broadcast of one or more disks using a broadcast channel
Each disk can be bcast at different speed
Disk speed can be changed based on client access pattern

Indexing on Air



- Server dynamically adjusts broadcast hotspot
- Clients read the index, enter into doze mode, and then perform *selective tuning*
  - *Query Time*: time taken from point a client issues a query until answer is received
  - *Listening Time*: time spent by client listening to the channel.

Caching in Mobile Client/Server: Problems

- Cashing is critical to many applications such as Web browsing and file and database systems
- Classical cache invalidation techniques do not work effectively in the mobile environment because of:
  - client disconnection (*call-backs* do not work)
  - slow and limited up-link bandwidth for client invalidation (*detection approach* is inefficient)
  - very limited scalability for application servers with a large number of clients
  - limited caching capacity due to client memory and power consumption limitations

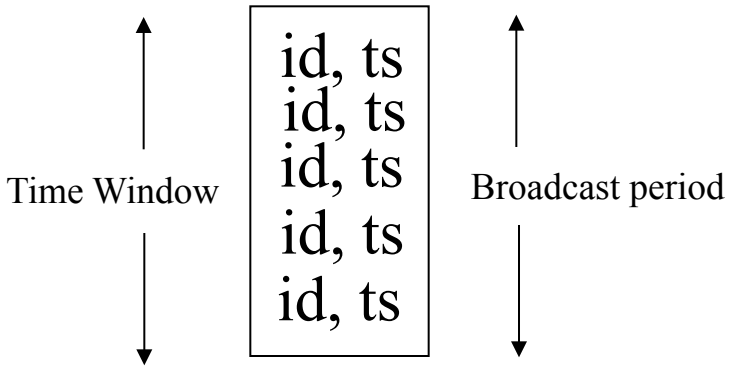
Caching in Mobile Client/Server: Solutions

- Variable granularity of cache coherency (Coda)
- Broadcast disks based cache invalidation

Varied Granularity of Cache Coherency in *Coda*

- Server maintains version stamps for each object and its containing volume. When an object is updated, the server updates its version stamp and that of its containing volume.
- In anticipation of disconnection, clients cache volume version stamps
- Upon reconnection, clients present volume version stamps to server for validation
- If valid, so is every object cached at the client, otherwise, cached objects are invalidated individually

BC Disk based Cache Invalidation
Cache Invalidation Reports



- Server bcast invalidation report showing all items updated within preceding *w* seconds
- Client connected: invalidation is straightforward
- Clients must invalidate their entire cache if their disconnection period exceeds *w* seconds.

Cloud-Mobile & Cyber Foraging Models

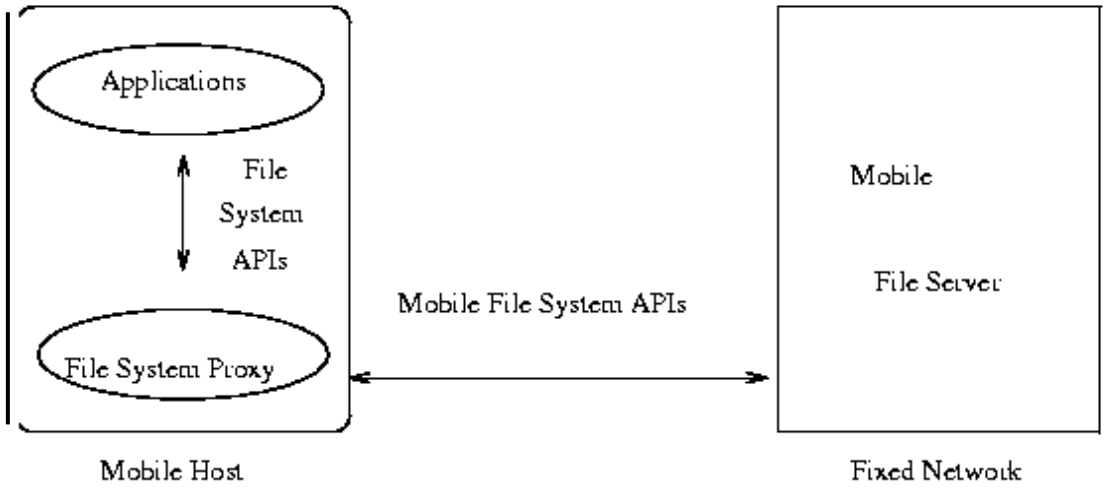
- Similar to thin client but it is not quite thin.
- Virtualized replica of the Mobile Platform (or client)
- Outsourced computations performed on the cloud replica using cloud resources – only results are synced.

Jason Flinn, “Cyber Foraging: Bridging Mobile and Cloud Computing,” Pervasive and Mobile Computing Synthesis lectures. September 2012 (pdf)

A. Abukmail and A. Helal, "Energy Management for Mobile Devices through Computation Outsourcing within Pervasive Smart Spaces," Submitted to the IEEE Transactions on Mobile Computing. Submitted January 2007. (pdf)

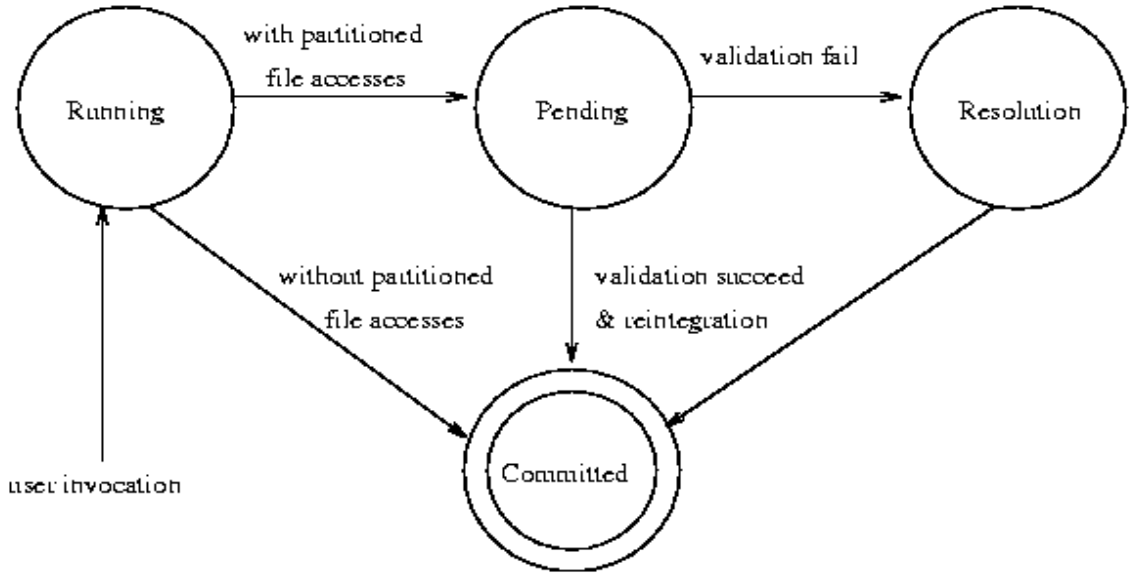
Case Studies

File System Proxy in *Coda*



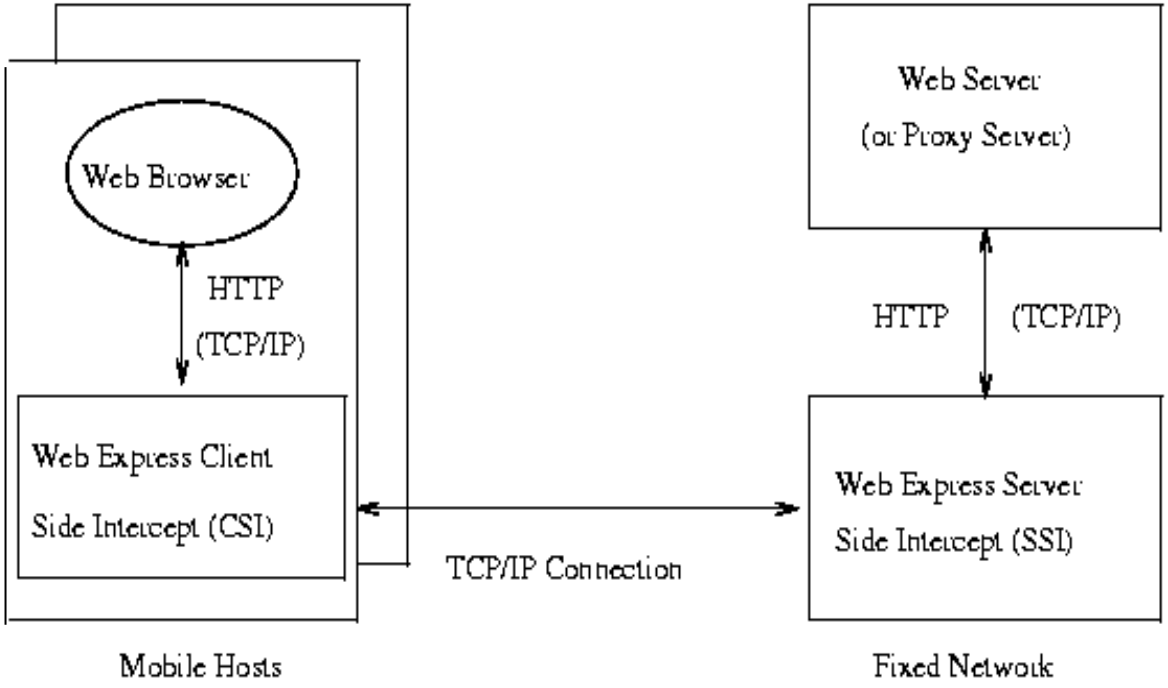
Disconnected operation (Venus): hoarding, emulating, reintegrating
Weakly connected operation: both object and volume call-backs
Isolation-Only Transactions

Isolation-Only Transactions in *Coda*



Isolation-Only Transactions (*ACID*): no failure atomicity guarantees. Also Durability is guaranteed only conditionally.

Web Proxy in *WebExpress*

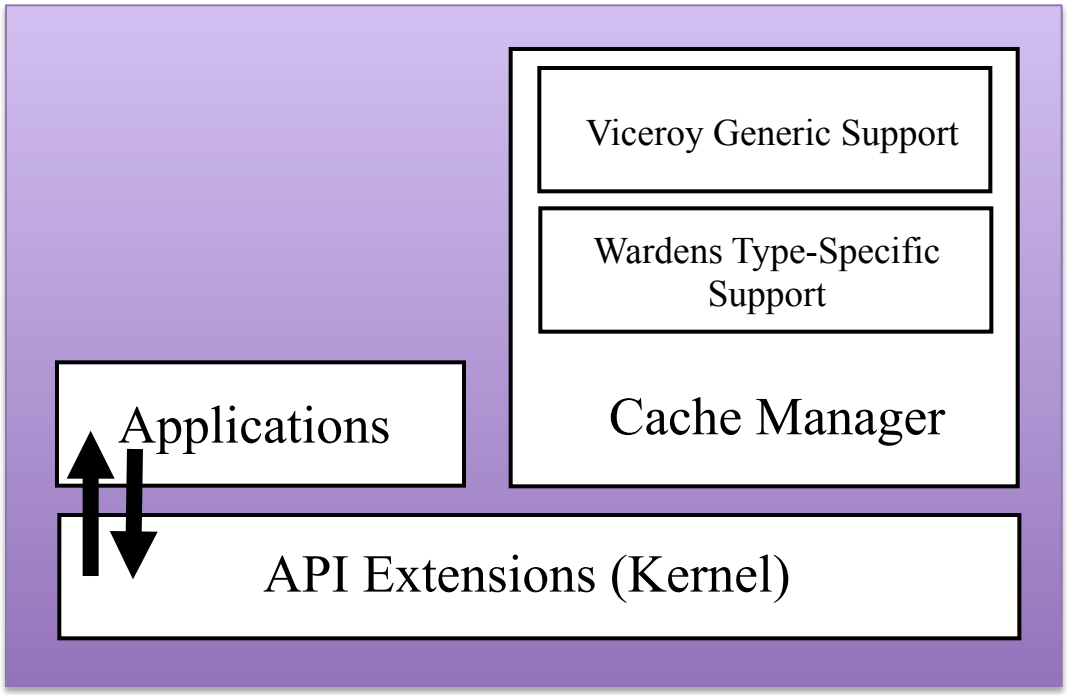


The WebExpress Intercept Model

Odyssey

- Odyssey client architecture
- Odyssey system components
- Odyssey applications:
  - Video player
  - Web browser

Odyssey Client Architecture



Main Features of Odyssey

- Application-aware adaptation approach
- Odyssey monitors system resources and notifies applications of relevant changes
- Applications decide when and how to adapt, to maintain certain level of fidelity
- General support for adaptation: *Viceroy*
- Type-specific support: *Warden*
- Caching support

Odyssey System Components

- Odyssey Objects
- Client API to allow applications to:
  - operate on Odyssey objects
  - express resource needs (expectations)
  - be notified when needed resources are no longer available
  - respond by changing levels of fidelity

Odyssey API

Request(in path, in resource_descriptor, out request_id)
Cancel(in request_id)

Resource Negotiation Operations

Resource_id
lower bound
upper bound
name of upcall handler

Resource Descriptor Fileds

Handle(in request_id, in resource_id, in resource-level)

Upcall Handler

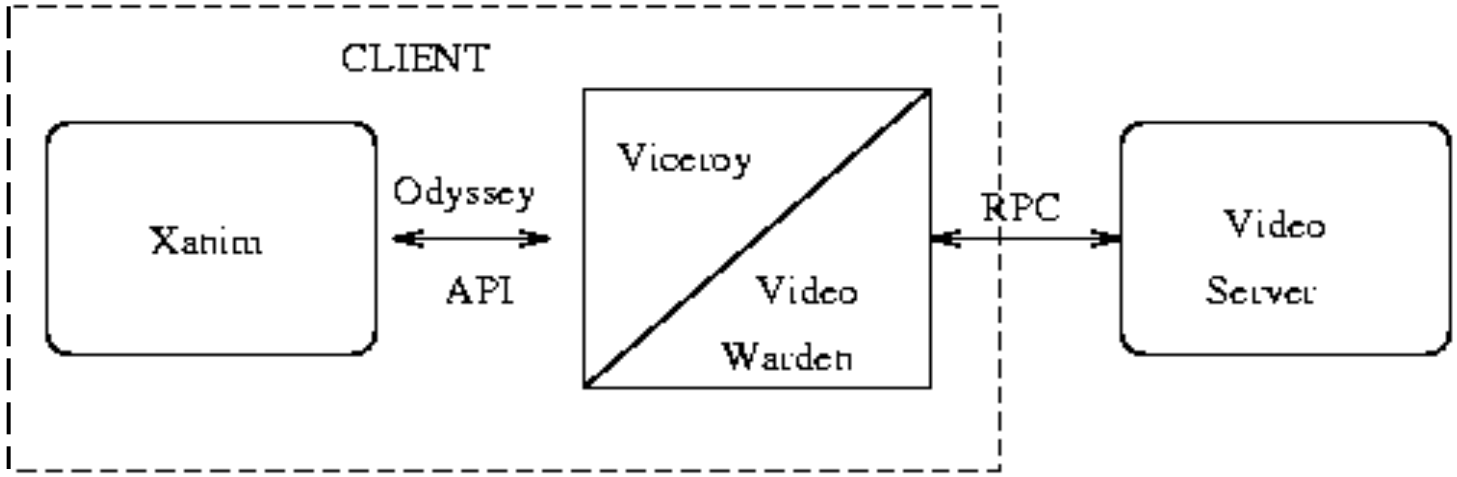
Network Bandwidth	*bytes/second*
Network Latency	*microseconds*
Disk cache Space	*Kilobytes*
CPU	*SPECCint95*
Battery Power	*minutes*
Money	*cents*

Generic Resources in Odyssey

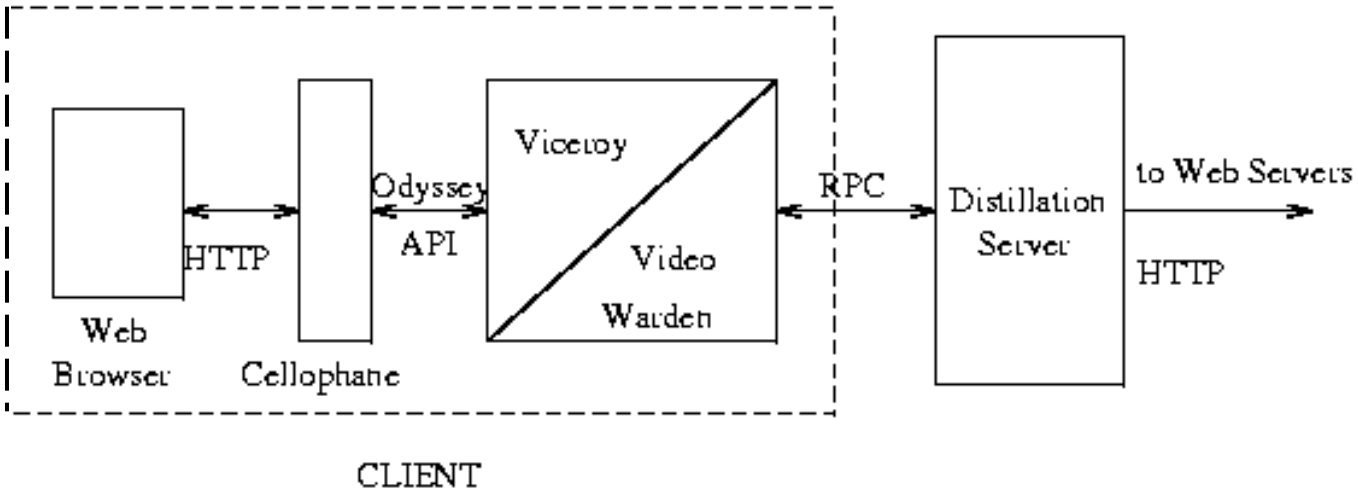
Tsop(in path, in opcode, in insize, in inbuf, in out outsize, out outbuf)

Type-specific Operations

Video Player in Odyssey



Web Browser in Odyssey



Odyssey: Summary

Applications	File system access, video playing, and Web browsing.
Adaptation	Application-aware adaptation with the system support that provides resource monitoring, notifies applications of resource changes, and enforces resource allocation decisions.
Model	Classic client-server architecture.
Mobile Data	Distilled server data delivery based on the client Feedbacks.

Mobile and Pervasive Computing

CNT5517 - Section 2D32 & CIS4930 - Section 2D78

Lecture 5

The Android Platform - I

Dr. Sumi Helal

Computer & Information Science & Engineering Department

University of Florida, Gainesville, FL 32611

helal@cise.ufl.edu

Lecture Outline

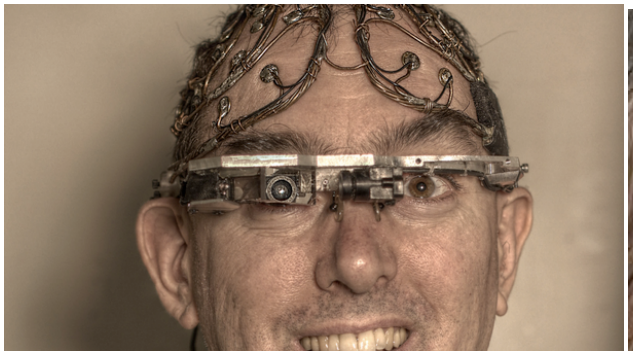
- Introduction
- Android Platform Architecture
- Android Application Frameworks
- Core Application Component Types
  - Activities
  - Content Provider
  - Services
  - Broadcast Receiver
- Component Interactions - Intents

What is Android?

an•droid |ˈanˌdroid|
noun
(in science fiction) a robot with a human appearance.

ORIGIN early 18th cent. (in the modern Lat form): from modern Latin ***androides***, from Greek ***anēr, andr- 'man'*** + -OID .

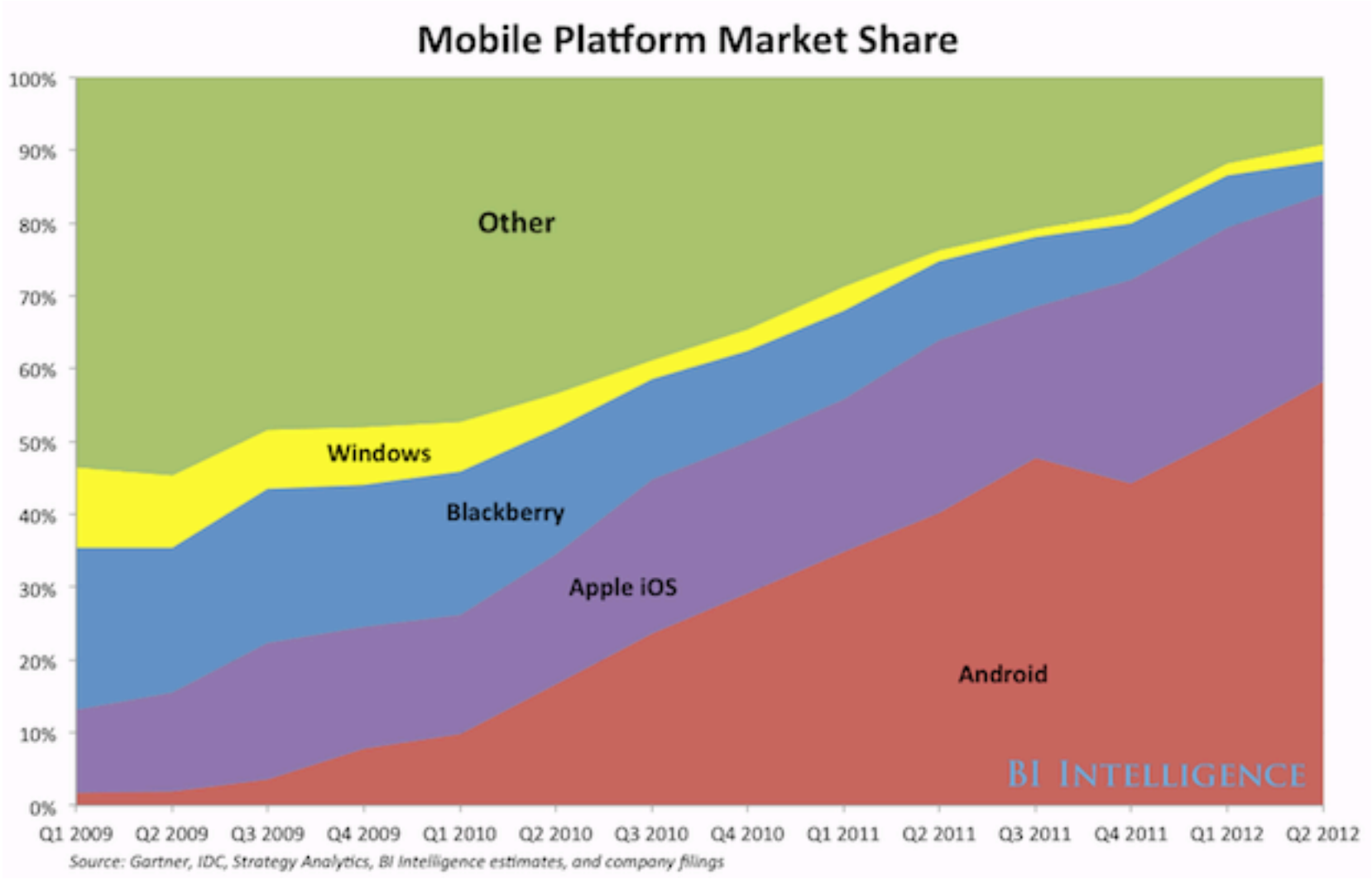
And what is a Cyborg?



From Humble Beginnings to Top Dog

- Small start up (Android, Inc., 1999) acquired by Google in 2005
- First version by Google in 2008
- Further developed by Open Handset Alliance
  - Device manufactures, chipset vendors, others
- Open source mobile platform under the GNU General Public License & Apache 2.0 license
- Some Apps are closed (e.g., Google Maps)

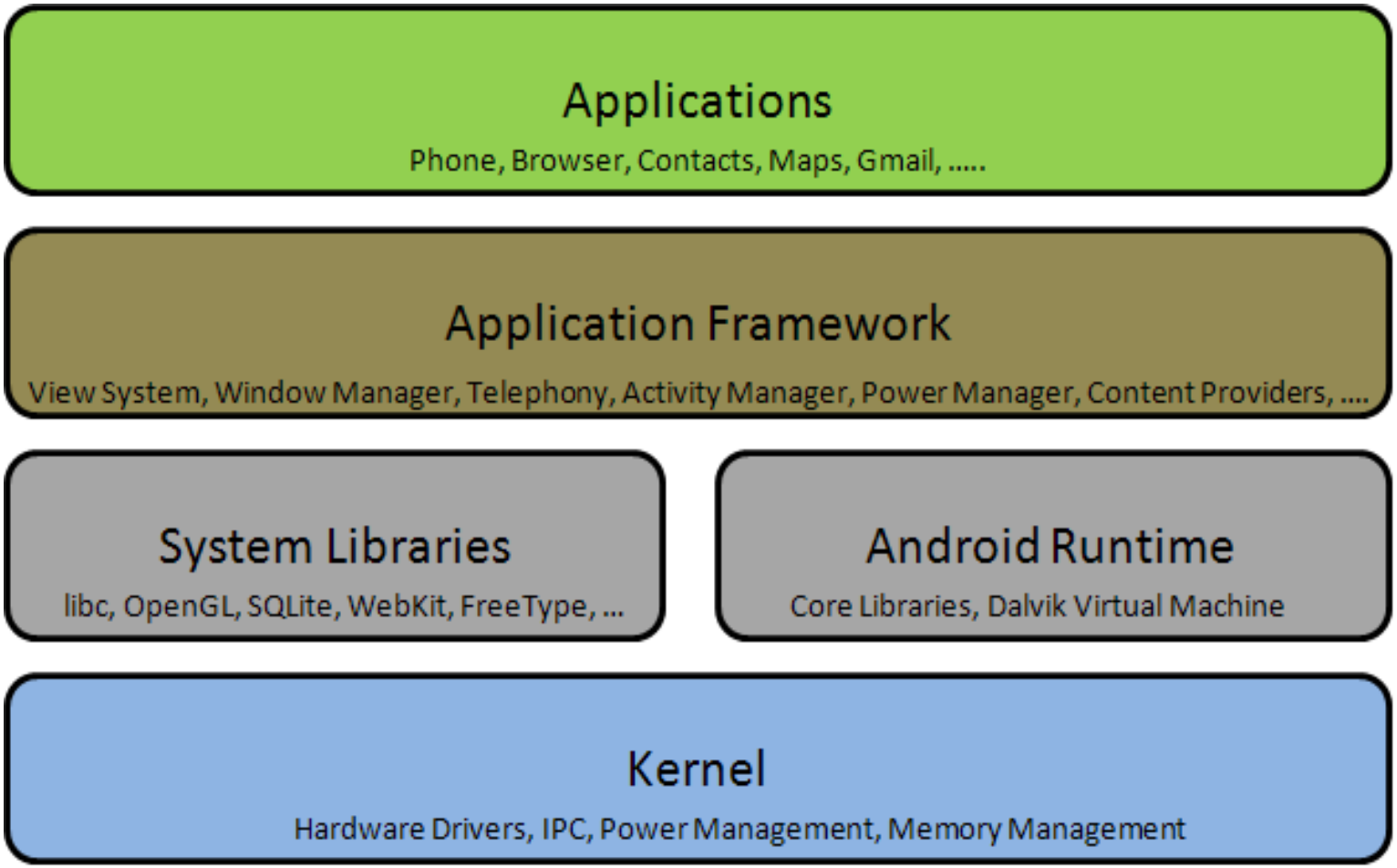
From Humble Beginnings to Top Dog



Platform Architecture

- Linux based (uses a heavily customized version of Linux kernel)
  - Does not include all GNU library
  - Has its own windowing system (no X Windows)
  - More
- Java based
  - A dialect and frame-worked version of Java (some but not all Java API)
  - Major differences in the UI libraries
  - Large developer base
  - Uses Eclipse IDE with Android Developer Tool plug-in

Platform Architecture



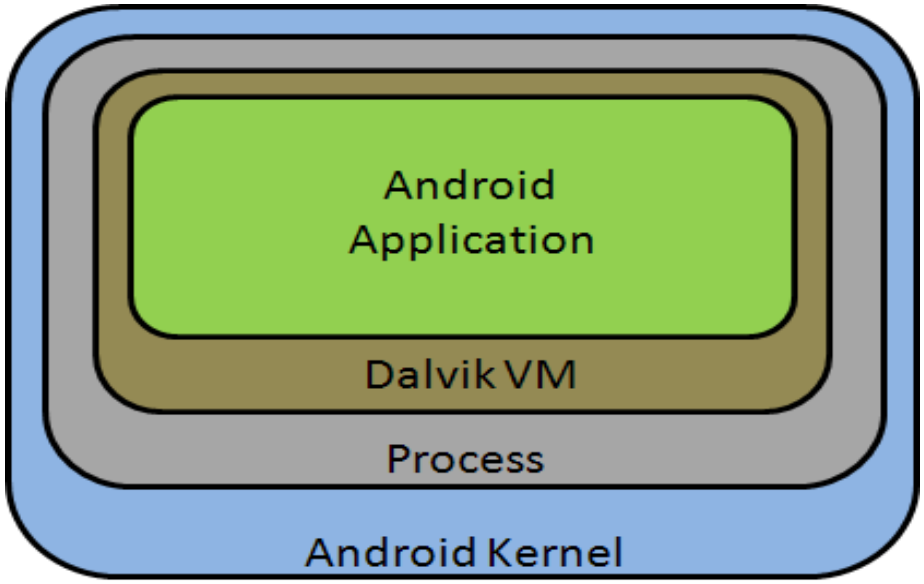
The Android Kernel

- Includes a HAL layer offering device drivers for display, multi-touch input, networking, power management and storage
- Own IPC (Binders)
  - Powerful IPC with support for handling shared objects lifecycle
- YAFFS2 (Yet Another Flash File System 2)
- WakeLock's – disables power down for UX

The Android Runtime System

- Dalvik Virtual Machine (VM): Core runtime component, atop the Android Kernel
- Dalvik: No Abstract Window Toolkit (AWT) support, no Swing.
- Android apps compiled to JVM compatible bytecode (.class), then converted to Dalvik executable (.dex) run by the Dalvik VM
- JIT compiler is also supported by Davlik

The Android Runtime System



- Each Android app executes within its own instance of the Dalvik VM which is, in turn, run as a kernel managed process.

Android System Libraries

- Not a complete set of standard GNU C libraries found in Linux
- Accessible through Application Frameworks
- Included: the C library (**libc**), **FreeType** (rendering text onto bitmaps), **SQLite** (databases), **OpenGL** for 2D and 3D rendering (utilizing hardware acceleration if available).
- **LibWebCore** library provides a WebKit* based browser engine which can be embedded as a web view within user interfaces of other applications. It is used in the development of the stock Android browser.

* WebKit is a layout engine software component for rendering web pages in web browsers. It powers Apple's Safari web browser and was previously used in Google's Chrome web browser. As of September 2013 WebKit browser market share[5] was larger than that of both the Trident engine used by Internet Explorer, and the Gecko engine used by Firefox.

Android System Libraries

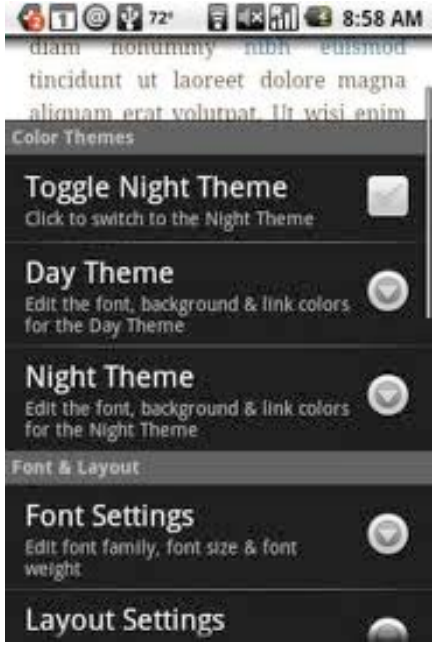
- The Android Media Library based on the OpenCORE multimedia framework developed by PacketVideo. Handles recording and playback of MP3, AAC, H.264, MP4, JPG and PNG.

Android Application Frameworks

- Application level APIs to utilize platform native capabilities in a simple yet powerful way
  - An empowerment for programmers
  - A required framework to do things right
- Main Frameworks:
  - View Manager
  - Content Provider
  - Notification Manager
  - Location Manager
  - Input Method Service
  - Telephony Manager
  - Power Manager

View Framework

- Key framework for creating UI
- Offers building blocks for rendering and managing events
- *View* Class
  - Any rectangular area is a View object
- *ViewGroup* Class
  - Extends View and corresponds to a full layout of the display.
  - A ViewGroup must have a corresponding XML layout



http://developer.android.com/resources/tutorials/views/index.html

Notification Framework

- Apps uses notification framework to notify the user about specific events.
- *NotificationManager* Class is used to:
  - allow an application to place an icon on the status bar
  - Turn backlight on
  - Vibrate
  - Play sounds

- In the icon case, when clicked by the user, app-specific actions are executed.

http://developer.android.com/reference/android/app/NotificationManager.html

Resource Manager Framework

- Allows programmers to decouple static resources and device specifics from the application code
  - Layouts, animations, strings, sounds, images, sizes, languages, etc.
- Leads to better app design and effective customization of the app to different devices.
- Each resource is accessed by addressing it using its package name, resource type, and resource name:
  - ***<package_name>.R.<resource_type>.<resource_name>***
    - where resource_type is set from a pre-determined list of values specified by Android.

http://developer.android.com/guide/topics/resources/index.html

Location Framework

- An enabler for (3rd party) LBS services
- *LocationManager* Class provides powerful locational services and at the same time abstracts away the details of location technology (GPS, WiFi triangulation, etc.)
- App developers use *LocationManager* to:
  - Identify user geographical location
  - Create triggers to execute actions when the user is in the proximity of a specific location

http://developer.android.com/reference/android/location/LocationManager.html

Input Method Framework

- The *InputMethodService* class enables developers to implement their own custom software keyboards, keypads and even pen input.
- The input is then converted into text and passed on to the target UI element.

http://developer.android.com/reference/android/inputmethodservice/InputMethodService.html

Telephony Framework

- The *TelephonyManager* class provides applications with the ability to determine telephony services on the devices and access specific subscriber information.
- Applications can also subscribe to telephony state changes
- The *SmsManager* allows applications to send data and text messages using the Short Message Service (SMS) protocol.
- Utilities: *PhoneNumberUtils* & *PhoneNumberFormattingTextWatcher* simplify the handling and formatting of phone number strings from around the world.

http://developer.android.com/reference/android/telephony/TelephonyManager.html

Power Management Framework

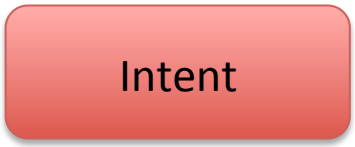
- The *PowerManager* class provides applications with the ability to control the power state of the device and use a feature called WakeLocks (defined by the *PowerManager.WakeLock* class) which forces a device to remain on and not go into power-saving mode.
- Enables better UX but risky and must be used cautiously
  - Only when absolutely necessary
  - Disable lock as soon as possible

http://developer.android.com/reference/android/os/PowerManager.html

Developing Android Applications

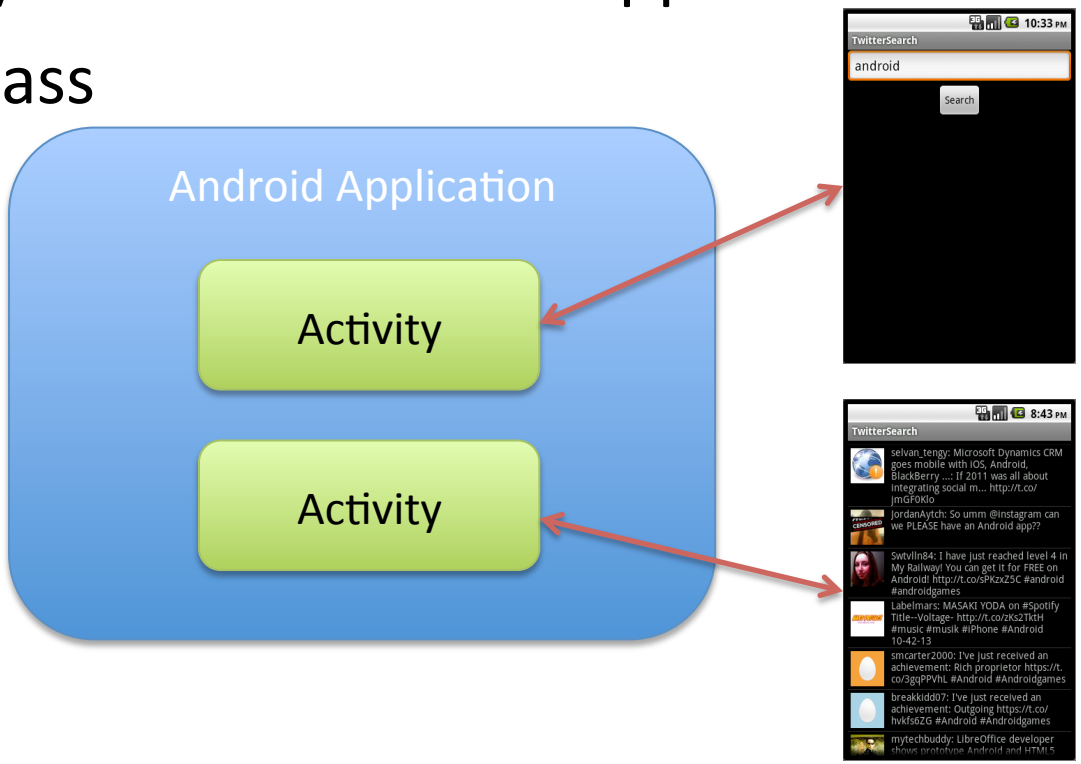
- Android Application is composed of 4 core types of components

- Interaction among Activities, Services ad Broadcast Receivers are done through a novel mechanism called Intent



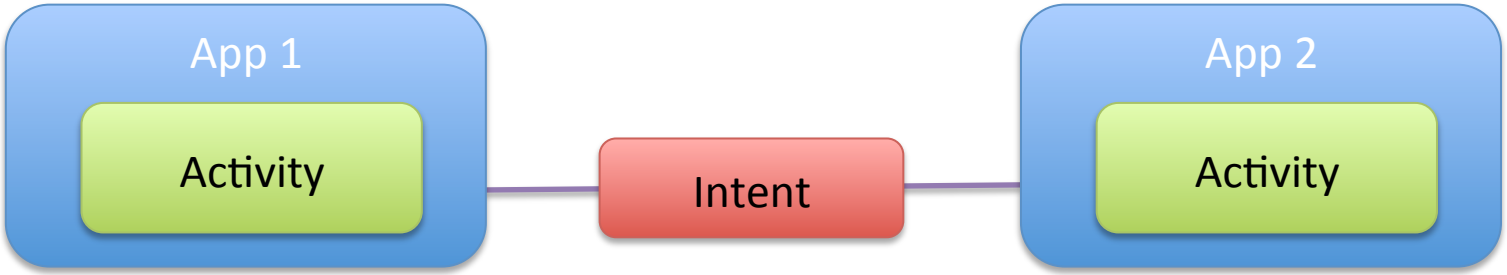
Activities

- An activity corresponds to a one of the application's UI.
  - An activity for each UI of the app
  - Activity Class



Activities

- Any activity may start another in another app by using Intents.
  - Example: email app may ask pdf viewer app to view a pdf attachment.
- An intent is an object that describes the intended request and any parameters passed
- Below: App 1 uses *startActivity (Intent intent)* to start a specific activity in App2.



Intents

- Example of an activity creating an intent and using it to start another

```
Intent i = new Intent(FirstrActivity.this, SecondActivity.class);
startActivity(i);
```

- *Explicit Intents* provide the exact class to be run in starting the target activity. They allow apps to launch various internal activities as the user interacts with the application.
- *Implicit Intents* only include enough information for the system to determine which of the available components are best to receive the intent (and in the case of activity, which should be started on receiving this intent). Requires *Intent Filters*.

Intent Objects

- An Intent object is a bundle of information. It contains information of interest to the component that receives the intent.
  - Action
    - A string naming the action to be performed.
  - Category
    - A string containing additional information about the kind of component that should handle the intent.
  - Data (parameters)
    - The URI of the data to be acted on and the MIME type of that data.

Predefined Action Constants

Constant	Target component	Action
`ACTION_CALL`	activity	Initiate a phone call.
`ACTION_EDIT`	activity	Display data for the user to edit.
`ACTION_MAIN`	activity	Start up as the initial activity of a task, with no data input and no returned output.
`ACTION_SYNC`	activity	Synchronize data on a server with data on the mobile device.
`ACTION_BATTERY_LOW`	broadcast receiver	A warning that the battery is low.
`ACTION_HEADSET_PLUG`	broadcast receiver	A headset has been plugged into the device, or unplugged from it.
`ACTION_SCREEN_ON`	broadcast receiver	The screen has been turned on.
`ACTION_TIMEZONE_CHANGED`	broadcast receiver	The setting for the time zone has changed.

Predefined Category Constants

Constant	Meaning
`CATEGORY_BROWSABLE`	The target activity can be safely invoked by the browser to display data referenced by a link — for example, an image or an e-mail message.
`CATEGORY_GADGET`	The activity can be embedded inside of another activity that hosts gadgets.
`CATEGORY_HOME`	The activity displays the home screen, the first screen the user sees when the device is turned on or when the *Home* button is pressed.
`CATEGORY_LAUNCHER`	The activity can be the initial activity of a task and is listed in the top-level application launcher.
`CATEGORY_PREFERENCE`	The target activity is a preference panel.

Intent-Filter

- To inform the system which implicit intents they can handle, activities, services, and broadcast receivers can have one or more intent filters.
- Each filter describes a capability of the component, a set of intents that the component is willing to receive. It, in effect, filters in intents of a desired type, while filtering out unwanted intents — but only unwanted implicit intents (those that don't name a target class).

Intent Filters

```
<intent-filter . . . >
    <action android:name="com.example.project.SHOW_CURRENT" />
    <action android:name="com.example.project.SHOW_RECENT" />
    <action android:name="com.example.project.SHOW_PENDING" />
    . . .
</intent-filter>
```

```
<intent-filter . . . >
    <category android:name="android.intent.category.DEFAULT" />
    <category android:name="android.intent.category.BROWSABLE" />
    . . .
```

```
<intent-filter . . . >
    <data android:mimeType="video/mpeg" android:scheme="http" . . . />
    <data android:mimeType="audio/mpeg" android:scheme="http" . . . />
    . . .
</intent-filter>
```

Example Intent Filter

- The Android system itself utilizes intent filters to populate the application launcher, the top-level screen that shows the applications that are available for the user to launch
- By finding all the activities with intent filters that specify both

  "android.intent.action.MAIN" action and

  "android.intent.category.LAUNCHER" category the

  Android system is able to display the icons and labels of those activities in the launcher.

Services

- Unlike Activities, Services are used for background, UI-less tasks that do not require user interactions.
  - Service Class
  - Examples: a background MP3 player, or an application profiler helping to identify which apps are most power hungry.
  - Similar to activities, start service using intents: *startService (Intent intent)*.

Services

- Each service must have a corresponding *<service>* declaration in its package's *AndroidManifest.xml.*
- Services can be started with *startService()*
- An application can start a service in another application by using *bindService()*. Bound services are servers in client-server interaction scenarios.
- Note that services, like other application objects, *run in the main thread of their hosting process*. This means that, if your service is going to do any CPU intensive work (such as MP3 playback) or blocking (such as networking) operations, it should spawn its own thread in which to do that work.

Content Provider

- One of the primary building blocks of Android applications, providing sharable content.
  - An application packages data as content provider
  - The application then shares the data with other applications through a single interface called the *ContentResolver* Interface.
- Example
  - Contacts are stored as content providers which allows them to be shared among other applications.
- If sharing is not intended, then storing data as content provider is meaningless. SQLite Database would be more appropriate in this case.

http://developer.android.com/guide/topics/providers/content-providers.html

Broadcast Receiver

- Tasked with responding to system-wide broadcast announcements.
- Android can broadcast a number of system status messages such as device battery status or when the camera has just captured a picture.
- Each broadcast is represented as an Intent object however it only contains the message associated with the broadcast and does not specify any Activity object that has to be triggered (Implicit Intent)

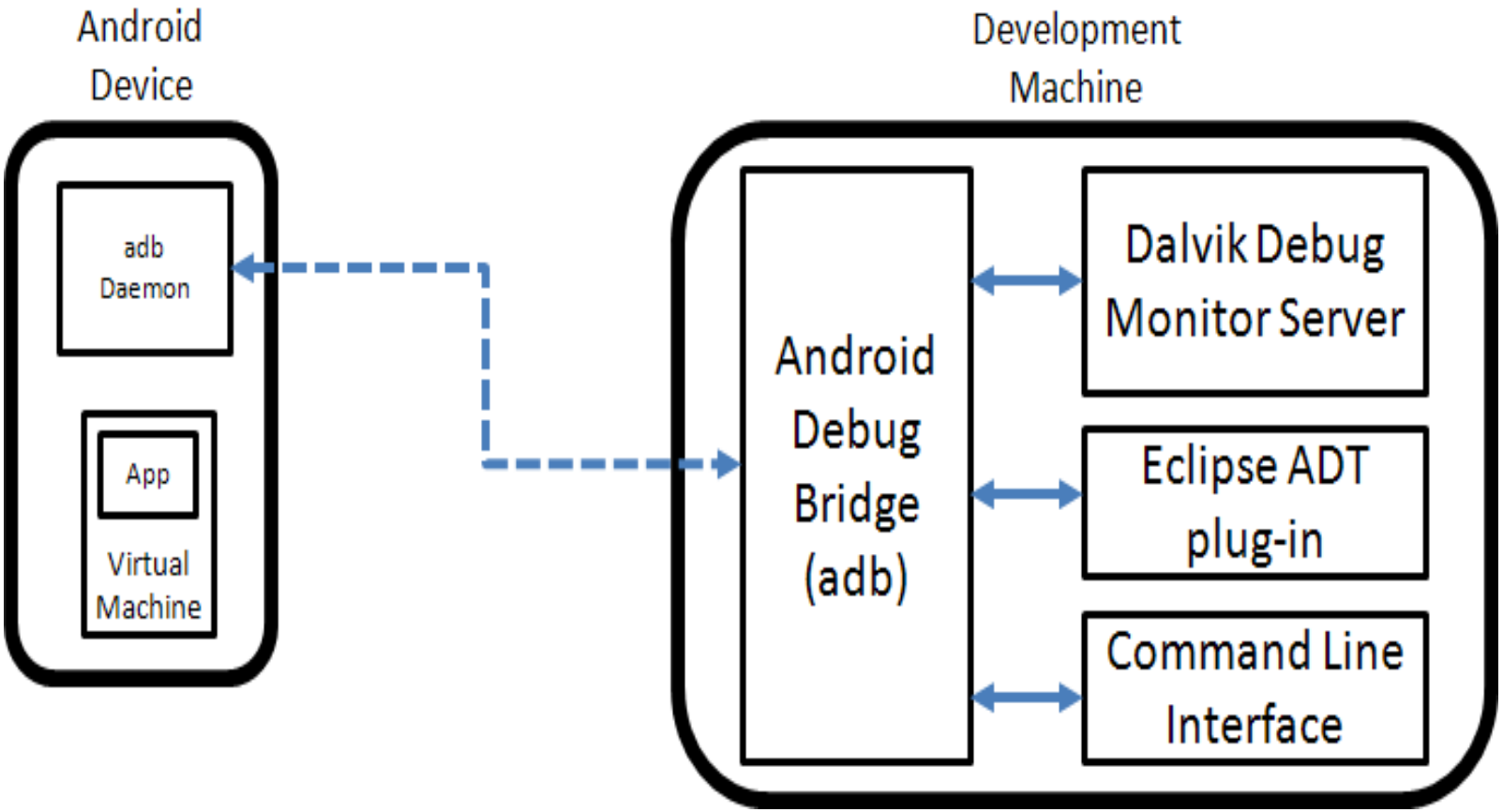
Android Software Development Kit

- We will use the Eclipse IDE as the development environment for Android Apps.
- An Android Development Tools (ADT) plug-in is available for Eclipse & can be downloaded from the Android developer web site.

Debugging and Testing

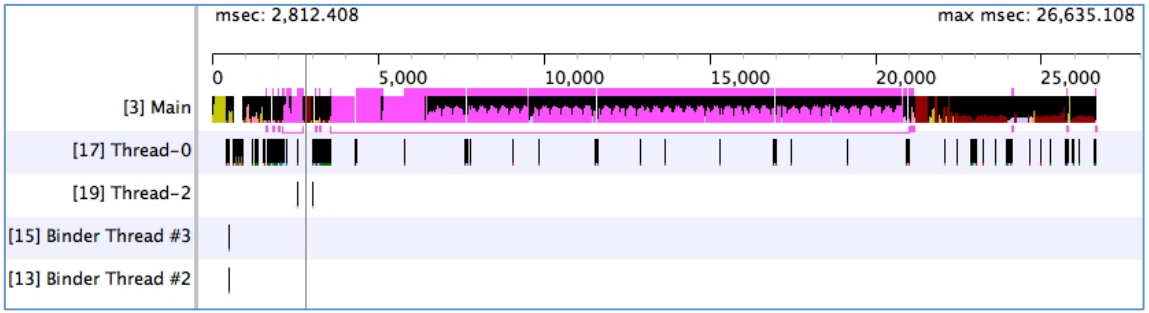
- Eclipse ADT comes with the Android Debug Bridge (adb) which allows for debugging apps running on
  - Actual devices, or
  - An Android Virtual Device (emulator)
- Dalvik Debug Monitor Server (DDMS) also comes with the ADT and interacts with the adb to show graphical information of running threads and call stack, among other features.

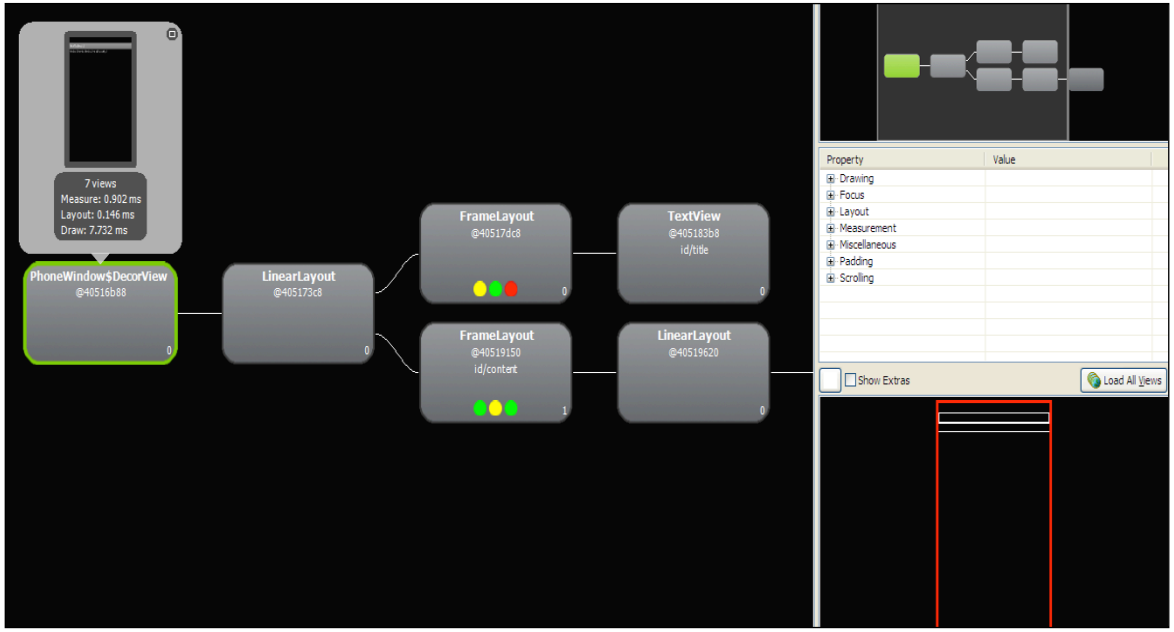
Debugging and Testing



Useful Tools

The Traceview Tool generates a graphical view of the application execution log





Hierarchy Viewer visualizes a layout of the the application views

Some Useful Android References

Open Handset Alliance Website	http://www.openhandsetalliance.com
Android Developers Website	http://developer.android.com
Android SDK Download	http://developer.android.com/sdk/
Eclipse Integrated Development Environment	http://www.eclipse.org/downloads/
Android API Reference	http://developer.android.com/reference/packages.html

References for Some Useful Android Components

Function	Component	Reference
User Interface Views and Layouts	ViewGroup	http://developer.android.com/resources/tutorials/views/index.html
Data Storage and Sharing	ContentProvider	http://developer.android.com/guide/topics/providers/content-providers.html
Notifications and Alerts	NotificationManager	http://developer.android.com/reference/android/app/NotificationManager.html
Application Resources and Localization	Resource Manager	http://developer.android.com/guide/topics/resources/index.html
Location-based Services	LocationManager	http://developer.android.com/reference/android/location/LocationManager.html
Customizing User Input	InputMethodService	http://developer.android.com/reference/android/inputmethodservice/InputMethodService.html
Phone Calls and SMS	TelephonyManager	http://developer.android.com/reference/android/telephony/TelephonyManager.html
Power Management	PowerManager	http://developer.android.com/reference/android/os/PowerManager.html

Mobile and Pervasive Computing

CNT5517 - Section 2D32 & CIS4930 - Section 2D78

Lecture 8

The Android Platform - II

Dr. Sumi Helal

Computer & Information Science & Engineering Department

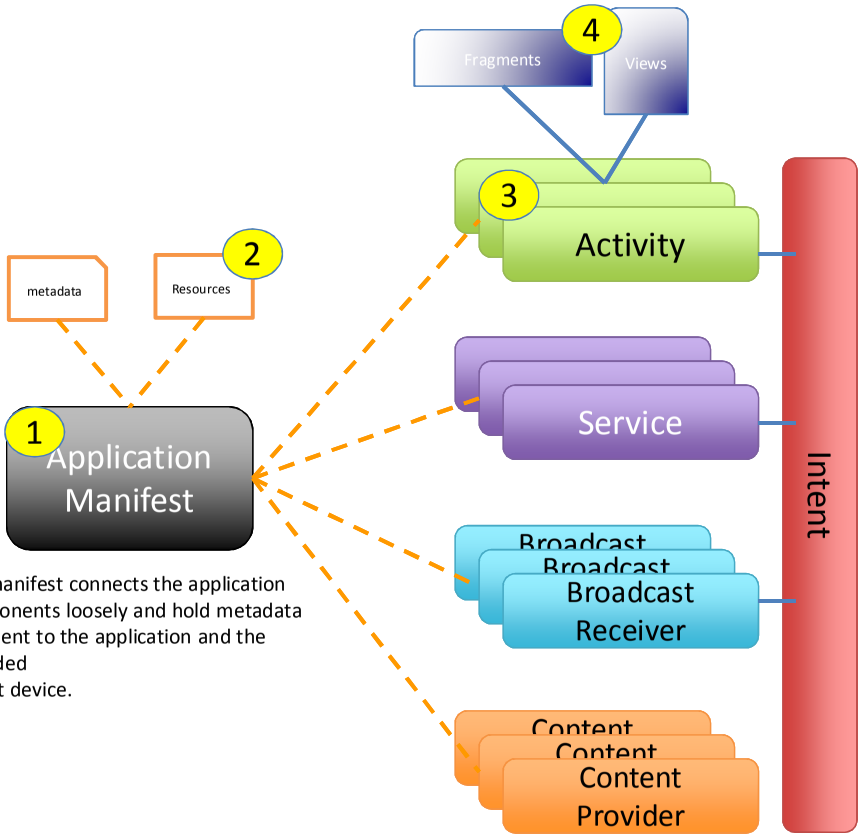
University of Florida, Gainesville, FL 32611

helal@cise.ufl.edu

Outline

- More on:
  - Manifest, Application Resources, Activities & Views
- The TwitterSearch Application
  - Design
  - Implementation

Android Application



The manifest connects the application Components loosely and hold metadata Pertinent to the application and the intended
Target device.

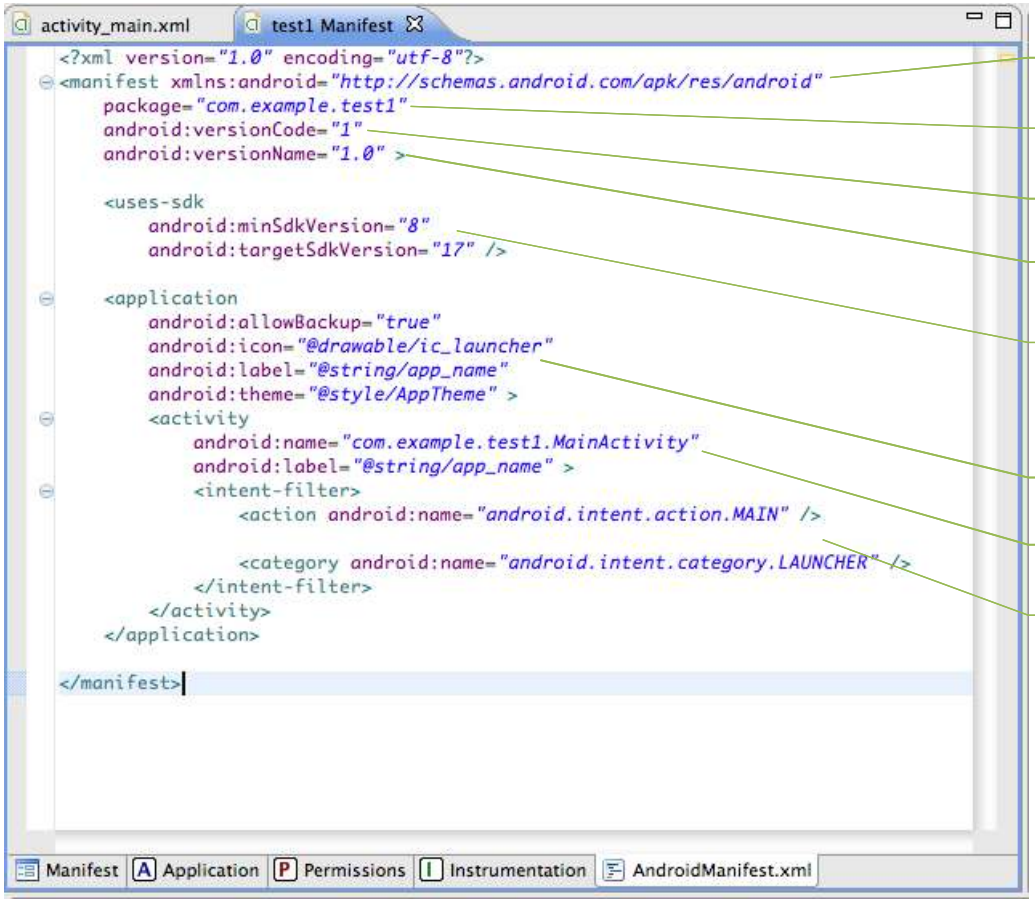
Activities – similar to Forms: layout and capture input.

Services - Long live computations, not requiring UI or user interactions.

Broadcast Receivers – Basically Intent listeners and processor.

Content Providers – sharable, persistent data objects, typically interact with SQL databases.

1 AndroidManifest.xml



Name Space ID & DTD location

Name of the Java Package

App version no.

App version name

Min: lowest version of SDK supported (lower => more devices).
Target: is highest version of SDK with which app was tested.

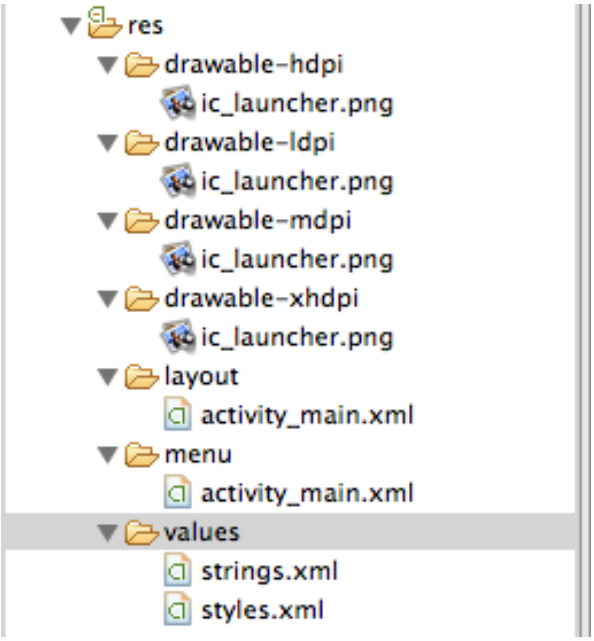
App meta data

Activities

Intent Filter

2 Application Resources

- Under res folder:
  - Generated automatically, extended, edited, …
  - Modular resources
- When application is built, all resources are compiled and compressed into the package.
- Also, the **R** class file containing all references to resources is generated. In the app, each reference to a resource uses **R**.
- Apps could access System resources as well.

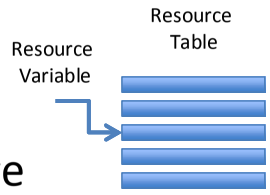


Using Resources

- Referring to Resources (Resource Variables)
  - ***<package_name>.R.<resource_type>.<resource_name>***
  - *In same package:* ***R.<resource_type>.<resource_name>***
  - Integer representing resource location in the resource table
  - Passing resource variables:
    - //inflate a layout resource

      setContentView(R.layout.test1);
    - //display a toaster with an error message

      Toast.makeText (this, R.string.app_error, …).show();



Using Resources

- Obtaining Resource Instances
  - Need to use helper classes to extract resources from resource table

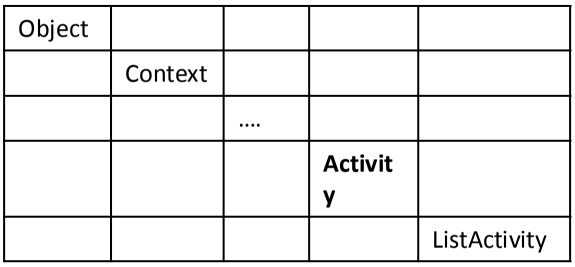
```
Resources myResources = getResources();
myResources.getText(R.string.stop_message);
myResources.getDrawable(R.drawable.app_icon);
myResources.getStringArray(R.array.string_array);
```

- Using System Resources
  - Handy native resources for all apps
  - Uses Android native Resource class (**android.R**) rather than the application class **R**.

```
getString(android.R.string.httpErrorBadUrl);
```

3 More on Activities

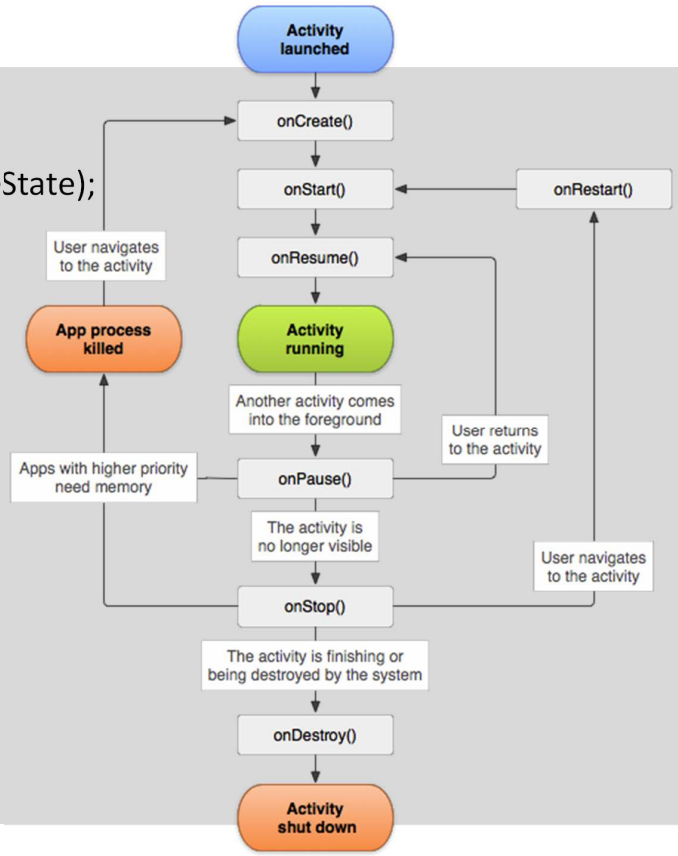
- Activity class takes care of creating a full screen to place a UI with setContentView(View) & implements some functionality.
- An Activity can be embedded inside of another using ActivityGroup.



- Each Activity must have a Layout
- Each Activity must be included in the Manifest file
- An Activity may have an Intent Filter.

Activity Life Cycle

```
public class Activity extends ApplicationContext {

    protected void onCreate(Bundle savedInstanceState);

    protected void onStart();

    protected void onRestart();

    protected void onResume();

    protected void onPause();

    protected void onStop();

    protected void onDestroy();
}
```

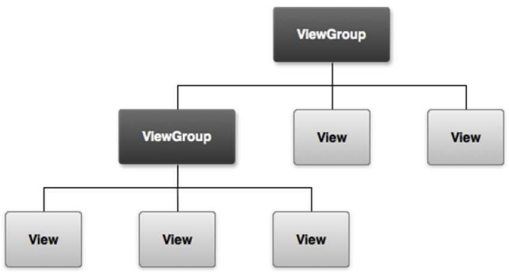


Use standard methods; Override to extend with additional tasks

4 Views

- View
  - Basic UI layout class
- ViewGroup
  - A View that contains other Views (Children Views)
    - ListView
    - GridView
    - SpinnerView
    - …

Object				
	View			
		View Group		
			Adapter View	
				ListView
				GridView
				SpinnerView
				…



setContentView

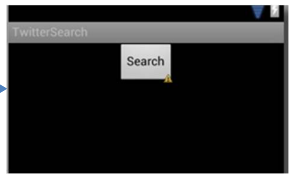
- A method of “Activity” class
- public void setContentView (View view)
  - Set the activity content to an explicit view. This view is placed directly into the activity's view hierarchy. It can itself be a complex view hierarchy. When calling this method, the layout parameters of the specified view are ignored.
  - Both the width and the height of the view are set by default to MATCH_PARENT.
    - To use your own layout parameters, invoke setContentView(android.view.View, android.view.ViewGroup.LayoutParams) instead.

LAYOUT_INFLATER_SERVICE

- Two ways to create user Interfaces
  - Procedural programming
  - Declarative xml file

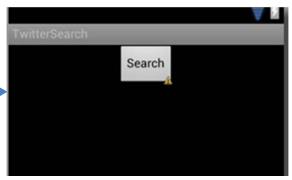
```
Button btn = new Button();
btn.setId("searchButton");
button.setText = "Search";
...
```

Created by ADT when compile & run




```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
   android:orientation="vertical"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent">
<Button
   android:id="@+id/searchButton"
   android:text="Search"
   android:layout_gravity="center"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"/>
</LinearLayout>
```

Context.LAYOUT_INFLATER_SERVICE
by inflater



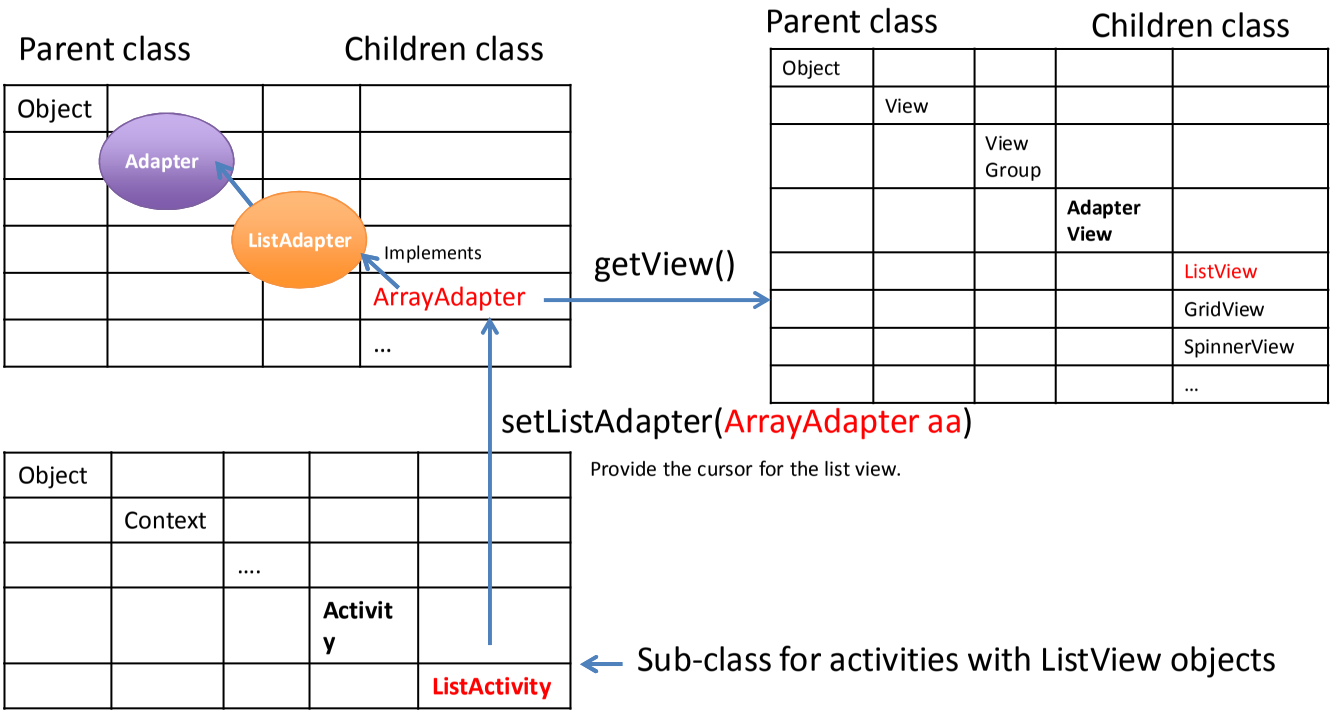

```
LayoutInflater inflater = (LayoutInflater)
context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
View tweetView = inflater.inflate(R.layout.tweet, parent, false);
```

List Adapters

- List Adapters
  - Used with ListView
  - They are Children View Controller (not Activities)
  - Bridge ListView and a Data Array of arbitrary object type
  - Use Inflate service to render children views populated with corresponding Array Data
  - There are other Adapters such as: GridAdapter, SpinnerAdapter, etc.

ListActivity-ArrayAdapter-ListView

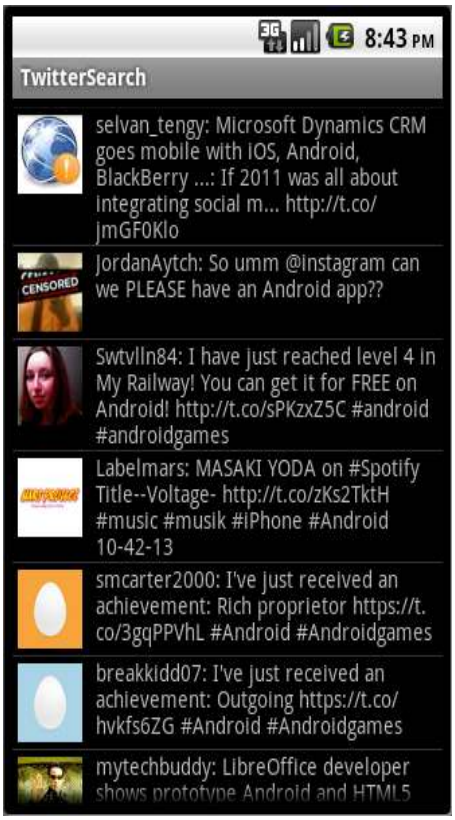
- Relationship bewteen ListActivity, ArrayAdapter, ListView

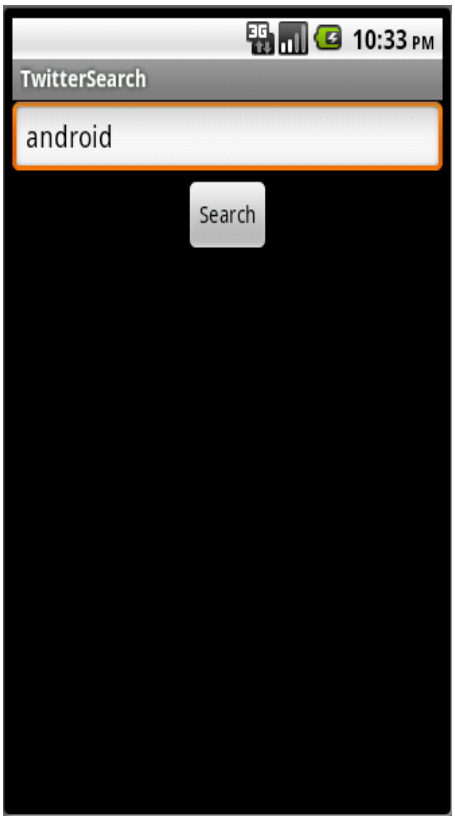


The TwitterSearch Application

- **Search Screen:** It contains one text input box which allows the user to input keywords to search for on Twitter, and a search button
- **Results Screen:** It presents the search results in a list, which shows the text of tweets containing the search item, the name of the tweet's author and the author's profile icon.

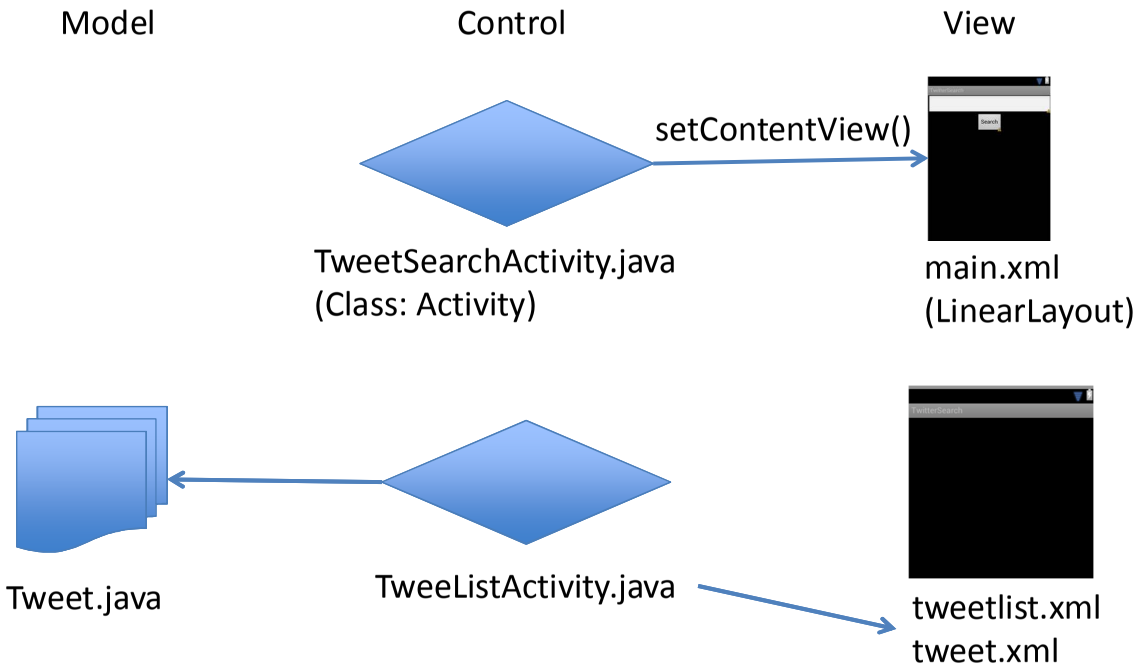
Android Search/Result Screens for Twitter Search





Android App Design

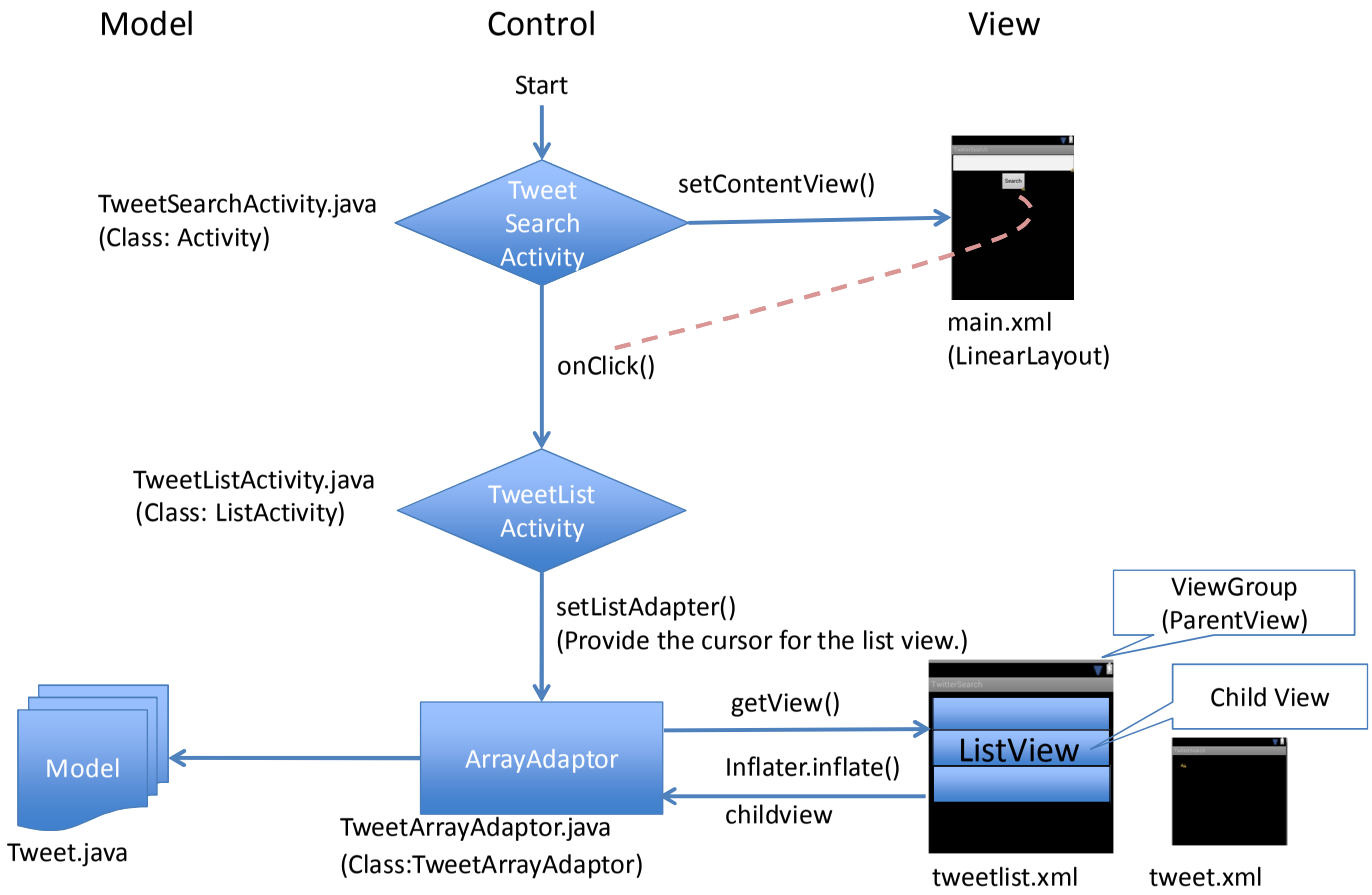
TwitterSearch Component Structure

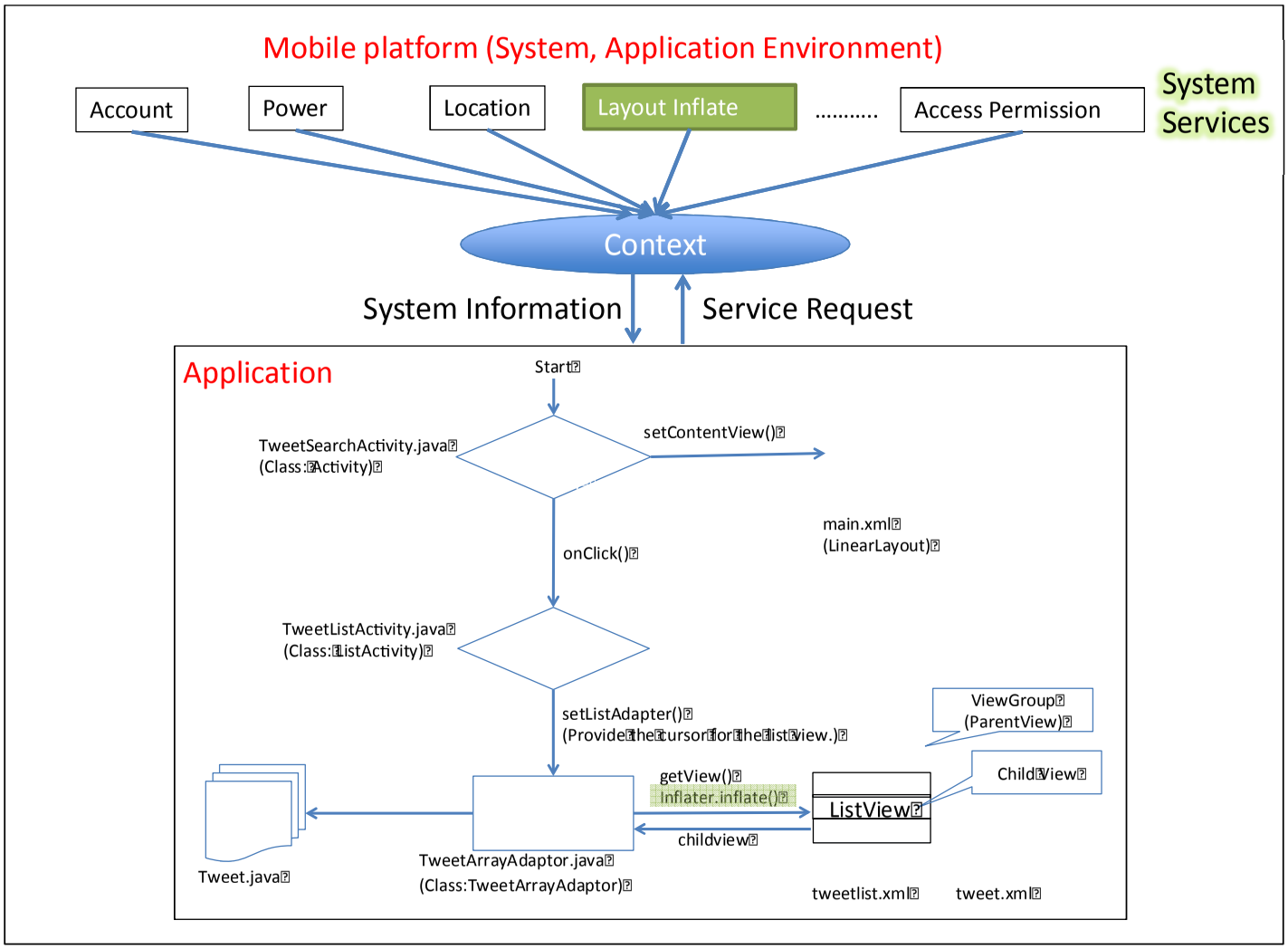


1 Model, 2 View, and 2 Control

Android App Design

TwitterSearch Control Flow



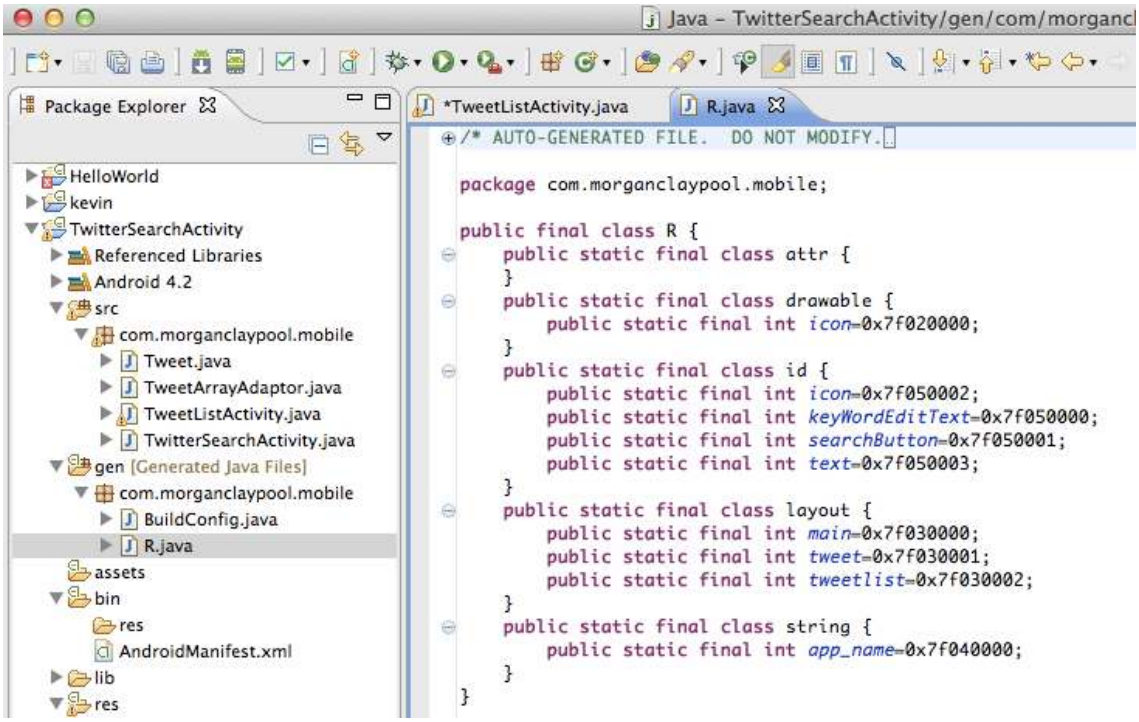

Mobile platform (System, Application Environment)
Account
Power
Location
Layout Inflate
.....
Access Permission
System Services
Context
System Information
Service Request
Application
Start
TweetSearchActivity.java
(Class:Activity)
setContentView()
main.xml
(LinearLayout)
onClick()
TweetListActivity.java
(Class:ListActivity)
setListAdapter()
(Provide the cursor for the listview.)
ViewGroup
(ParentView)
ChildView
getView()
inflater.inflate()
ListView
childview
Tweet.java
TweetArrayAdaptor.java
(Class:TweetArrayAdaptor)
tweetlist.xml
tweet.xml

TwitterSearchActivityManifest.xml

TwitterSearchActivity Manifest

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="com.morganclaypool.mobile"
      android:versionCode="1"
      android:versionName="1.0">
  <application android:icon="@drawable/icon" android:label="@string/app_name">
    <activity android:name=".TwitterSearchActivity"
              android:label="@string/app_name">
      <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
      </intent-filter>
    </activity>
    <activity android:name=".TweetListActivity"></activity>
  </application>
  <uses-permission android:name="android.permission.INTERNET"></uses-permission>
  <uses-sdk android:minSdkVersion="8" />
</manifest>
```

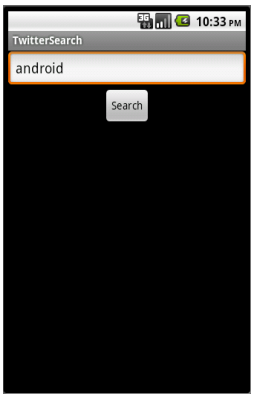
The R Resource Class



Layout: main.xml

main.xml

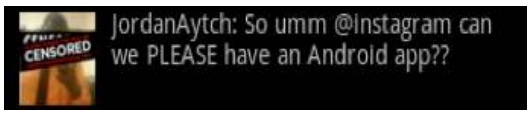
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
<EditText
    android:id="@+id/keyWordEditText"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"/>
<Button
    android:id="@+id/searchButton"
    android:text="Search"
    android:layout_gravity="center"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"/>
</LinearLayout>
```



Layout: tweet.xml

main.xml | tweet.xml

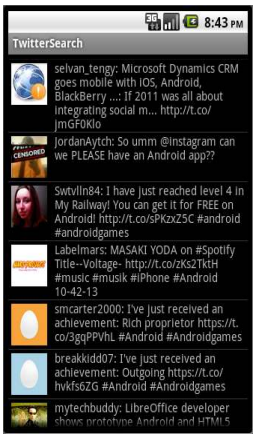
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" >
<ImageView
    android:id="@+id/icon"
    android:layout_width="48px"
    android:layout_height="48px"
    android:layout_marginLeft="4px"
    android:layout_marginRight="10px"
    android:layout_marginTop="4px"
    android:layout_marginBottom="4px">
</ImageView>
<TextView
    android:id="@+id/text"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content">
</TextView>
</LinearLayout>
```



Layout: tweetlist.xml

activity_main.xml | tweet.xml | tweetlist.xml

```
<?xml version="1.0" encoding="utf-8"?>
<ListView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
</ListView>
```



Tweet.java

Tweet.java

```
package com.morganclaypool.mobile;

public class Tweet {

   public String user;
   public String text;
   public String iconUrl;

}
```

TweeterSearchActivity.java

Tweet.java | TwitterSearchActivity.java

```
package com.morganclaypool.mobile;

import android.app.Activity;

public class TwitterSearchActivity extends Activity {

  @Override
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    final Button searchButton = (Button) findViewById(R.id.searchButton);
    searchButton.setOnClickListener(new View.OnClickListener() {
      public void onClick(View view) {
        Intent intent = new Intent();
        intent.setClass(TwitterSearchActivity.this, TweetListActivity.class);

        Bundle bundle = new Bundle();
        final EditText keywordEditText = (EditText) findViewById(R.id.keyWordEditText);
        bundle.putString("keyword", keywordEditText.getText().toString());
        intent.putExtras(bundle);

        startActivity(intent);
      }
    });
  }

}
```

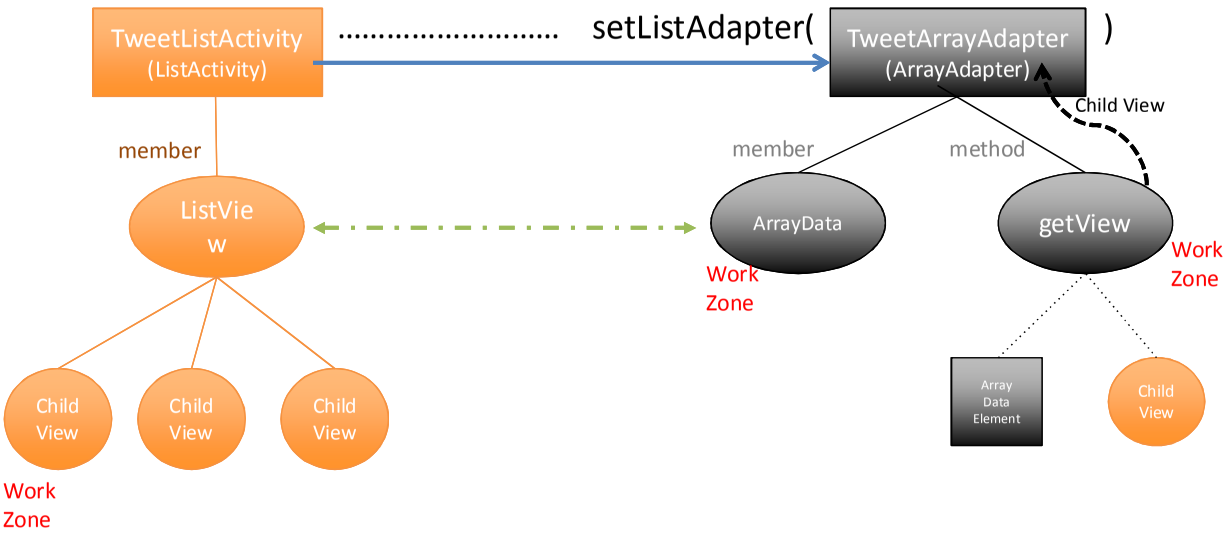
TweetListActivity.java


```
package com.morganclaypool.mobile;

import java.io.*;

public class TweetListActivity extends ListActivity {

  private static final String TWITTER_SEARCH_API = "http://search.twitter.com/search.json?lang=en&q=";
  private DefaultHttpClient httpClient = new DefaultHttpClient();

  @Override
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    Bundle bundle = this.getIntent().getExtras();
    String keyword = bundle.getString("keyword");

    TweetArrayAdaptor adapter = new TweetArrayAdaptor(this, searchTweets(keyword));
    setListAdapter(adapter);
  }

  private List<Tweet> searchTweets(String keyword) {
    List<Tweet> tweetList = new ArrayList<Tweet>();
    String resultString = get(TWITTER_SEARCH_API + URLEncoder.encode(keyword));
    try {
      JSONArray resultJsonArray = (new JSONObject(resultString)).getJSONArray("results");
      JSONObject jsonObject = null;
      for (int i = 0; i < resultJsonArray.length(); i++) {
        jsonObject = resultJsonArray.getJSONObject(i);
        Tweet tweet = new Tweet();
        tweet.user = jsonObject.getString("from_user");
        tweet.text = jsonObject.getString("text");
        tweet.iconUrl = jsonObject.getString("profile_image_url");
        tweetList.add(tweet);
      }
    } catch (JSONException e) {
      e.printStackTrace();
    }
    return tweetList;
  }
```

TweetListActivity.java

```
  private String get(String url) {
    String responseMessage = null;
    HttpGet httpGet = new HttpGet(url);
    try {
      HttpResponse getResponse = httpClient.execute(httpGet);
      HttpEntity getResponseEntity = getResponse.getEntity();
      if (getResponseEntity != null) {
        responseMessage = EntityUtils.toString(getResponseEntity);
      }
    } catch (IOException e) {
      httpGet.abort();
      e.printStackTrace();
    }
    return responseMessage;
  }

}
```

List Adapter in Action



TweetArrayAdaptor.java

Tweet.java | TweetArrayAdaptor.java

```
package com.morganclaypool.mobile;

import java.io.*;

public class TweetArrayAdaptor extends ArrayAdapter<Tweet> {

  private final Context context;
  private final List<Tweet> tweets;

  public TweetArrayAdaptor(Context context, List<Tweet> tweets) {
    super(context, R.layout.tweet, tweets);
    this.context = context;
    this.tweets = tweets;
  }

  @Override
  public View getView(int position, View convertView, ViewGroup parent) {
    LayoutInflater inflater = (LayoutInflater) context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
    Tweet tweet = tweets.get(position);
    View tweetView = inflater.inflate(R.layout.tweet, parent, false);
    TextView textView = (TextView) tweetView.findViewById(R.id.text);
    textView.setText(tweet.user + ": " +tweet.text);
    ImageView imageView = (ImageView) tweetView.findViewById(R.id.icon);
    imageView.setImageDrawable(loadImageFromURL(tweet.iconUrl));
    return tweetView;
  }

  private Drawable loadImageFromURL(String url) {
    Drawable drawable = null;
    try {
      InputStream is = (InputStream) new URL(url).getContent();
      drawable = Drawable.createFromStream(is, "srcname");
    } catch (IOException e) {
      e.printStackTrace();
    }
    return drawable;
  }

}
```

Mobile and Pervasive Computing
CNT5517 - Section 2D32 & CIS4930 - Section 2D78
Lecture 7

Introduction to Pervasive Computing

Dr. Sumi Helal
Computer & Information Science & Engineering Department
University of Florida, Gainesville, FL 32611
helal@cise.ufl.edu

The Computer Evolution

Mainframe Computer, 1960

The PC, 1980

Mobile Computer 1990

Sensor Platforms 2000

Smart Dust …

Sensor Network Forecast

- “The Quest for the Next Big Thing”
  - Business Week, August 2003
    - Utility Computing
    - The Sensor Revolution
    - Plastic Electronics
    - Bionic Bodies

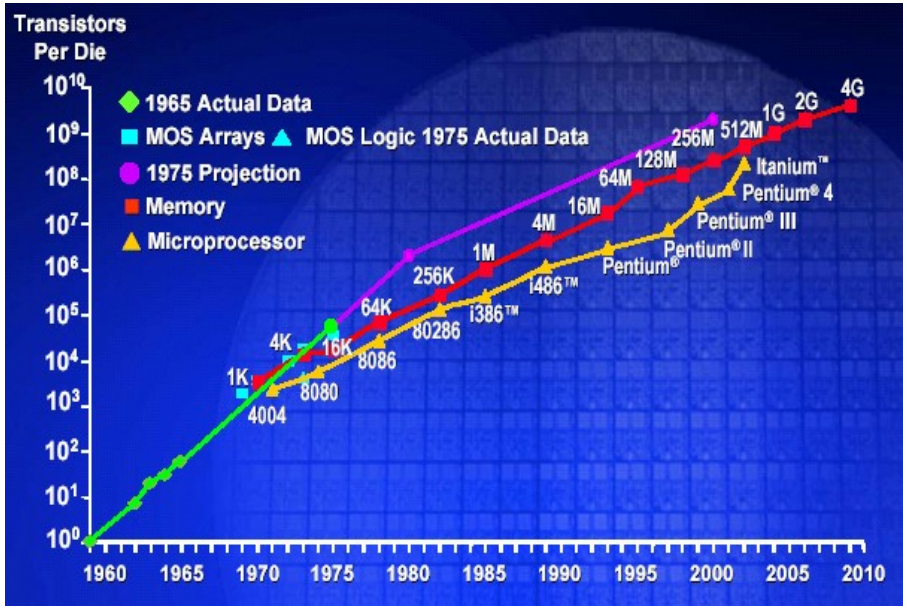
- “10 Emerging Technologies That Will Change The World”
  - MIT ENTERPRISE TECHNOLOGY REVIEW, Feb. 2003
    - Brain-Wireless Sensor Networks
    - Grid computing
    - Software Assurance
    - and more

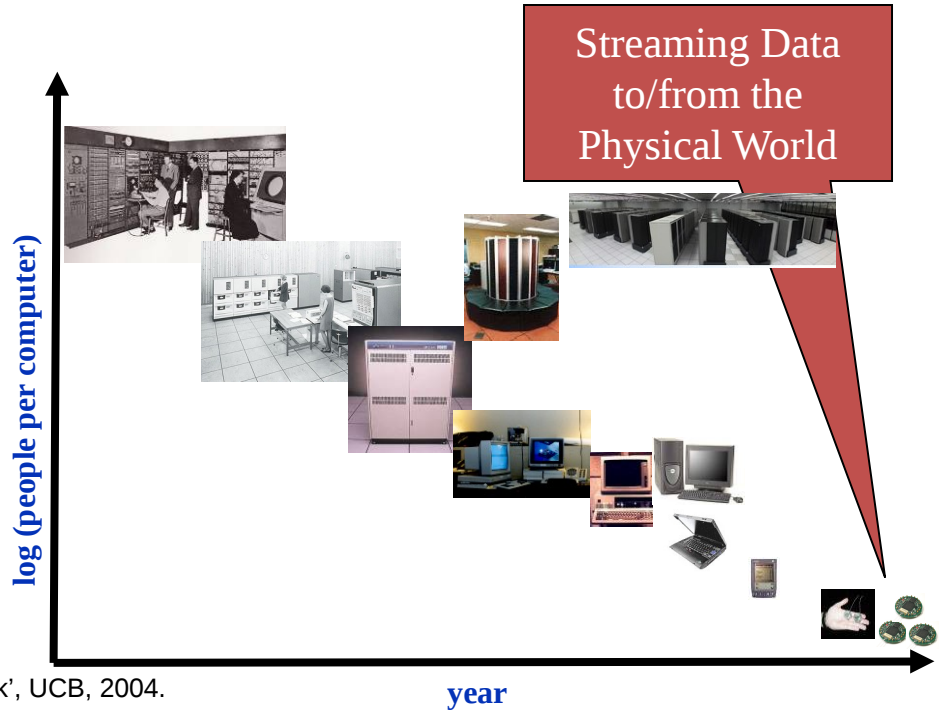
AN MIT ENTERPRISE
TECHNOLOGY
REVIEW
BUSINESS · OPPORTUNITY · IMPACT

Trend – Faster, Smaller, Numerous

- Moore's Law
  - "Stuff" (transistors, etc) doubling every 1-2 years

- Bell's Law
  - New computing class every 10 years



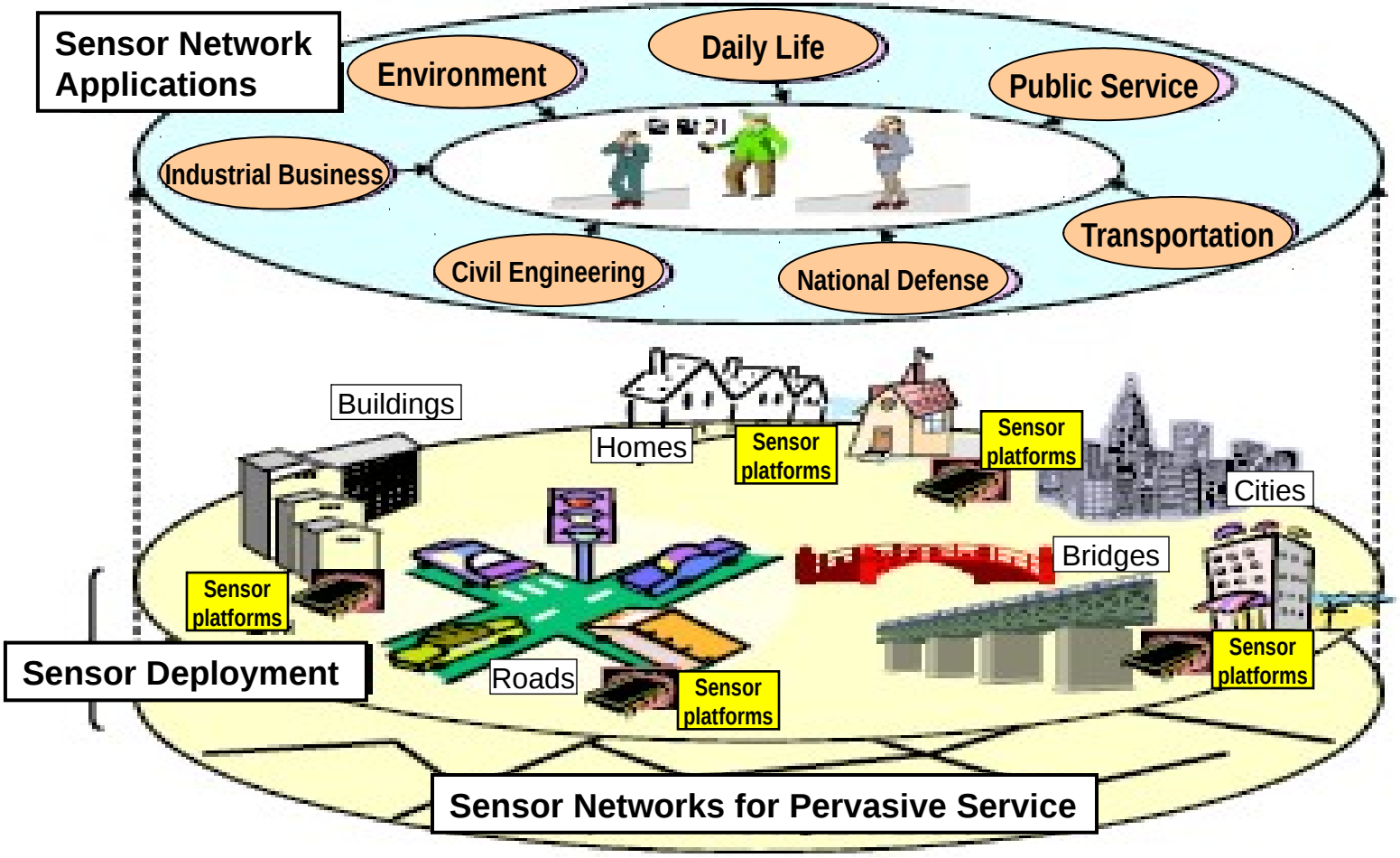


Excerpted from 'The Mote Revolution: Low Power Wireless Sensor Network', UCB, 2004.

- “Sensor Nation”, IEEE Spectrum magazine, July 2004

We will soon be able to know almost everything about everyone.

Promising Applications



Excerpted and modified from ‘Ubiquitous City Plan’,NCA, Korea, 2005.

Promising Applications

Daily Life

- Home Automation
- Home Security
- Sports & Fitness
- Weather Forecasting
- Childcare / Baby monitoring
- Elderly remote monitoring
- Local Community Activities

Public Service

- Energy Saving
- Water Resource Saving
- Smart Schools
- Smart Learning
- Medical Service
- Rehabilitation
- Medical Surgery/Treatment
- Healthcare & Immunization
- Postal Service
- Governmental Service

Transportation

- Traffic Monitoring
- Traffic Accident Avoidance
- Transportation Traceability
- Connectivity Optimization
- Shortest Routing Service
- Logistic & Delivery
- Vehicle Sensors
- Tire Pressure Real-time Monitoring sensors
- Roadside sensor deploying

Environment

- Monitoring Ocean Pollutant
- Monitoring Terrestrial Habitants
- Natural Disaster Avoidance
- Foods Traceability
- Agriculture Automation
- Poultry & Meat Traceability
- Man Disaster Avoidance
- Integrated Biology
- Habitant Monitoring

National Defense

- Military Operation Assist
- Military Resource Management
- Personnel Management
- Tactics & Battlefield Assist
- Logistic Traceability
- Frontier Guard Assist
- Open public space Surveillance (airports…)
- Mobile C4I Services

Industrial Business

- Sales Market Monitoring
- Logistics and Delivery
- Office Automation
- Manufacturing Automation
- Factory Automation
- Building Automation
- Legacy SCM, CRM and ASP Interface
- Resources Sensing (underground resources)

Civil Engineering

- Logistic
- Bridge Health Monitoring
- Architecture Monitoring
- Structural Monitoring
- Engineering Measurement
- Tension & Cracks Monitoring
- Road Monitoring
- Corrosion Monitoring

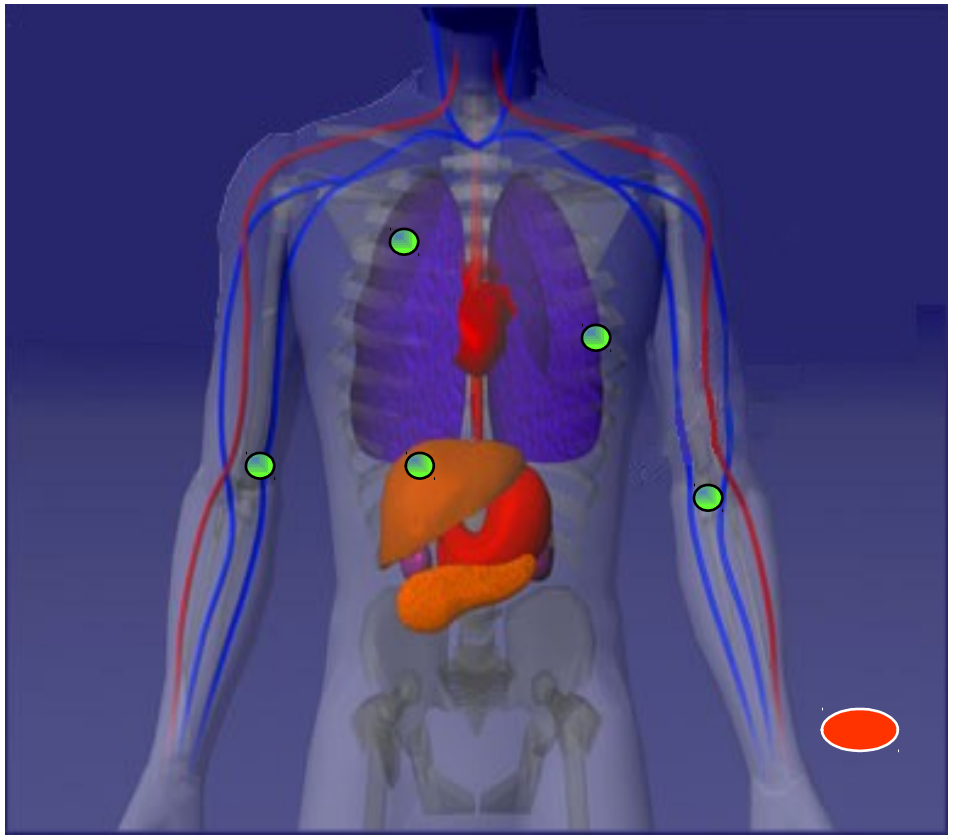
Daily Life Applications

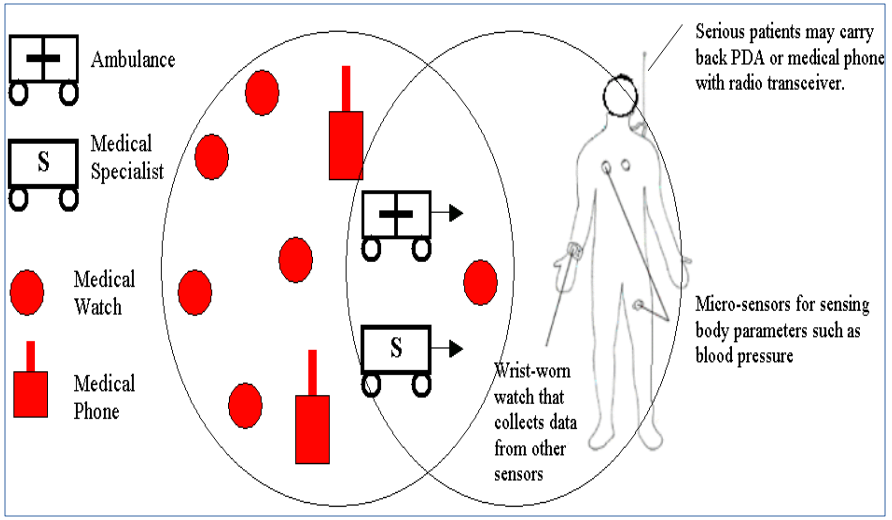
- Home Automation

Smart Grocery Shopping

Public Service Applications
- Medical Service

- Healthcare: Embedded Sensor networking for patient monitoring

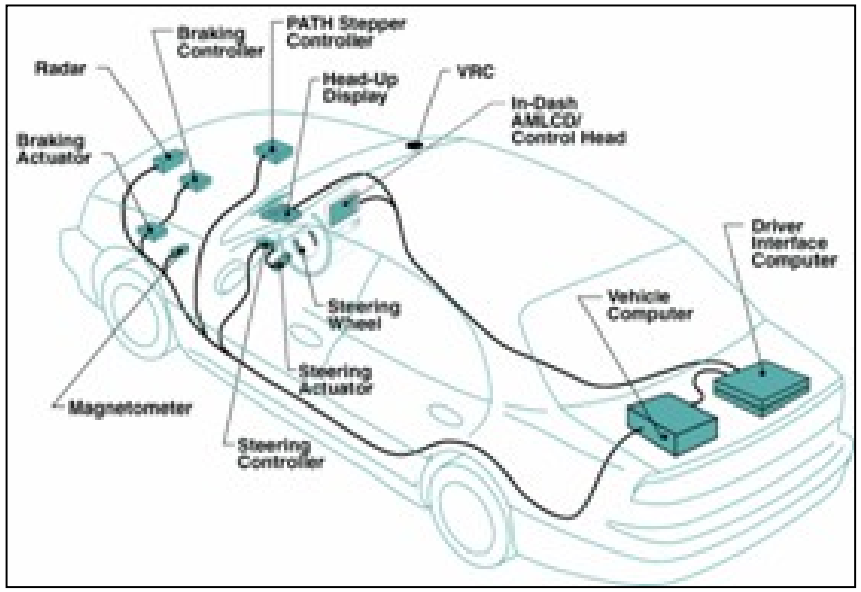




Excerpted from ‘Sensor Network Survey’, ICU, 2005.

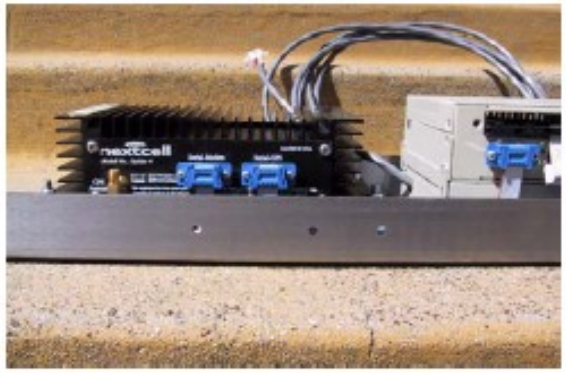
Transportation Applications
- Vehicle Sensors

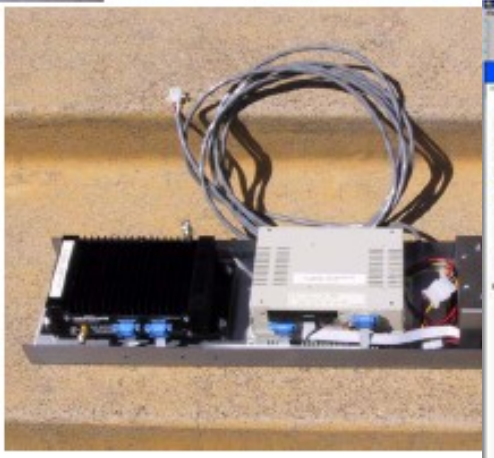
- Smart Cars?
  - Safety
  - Power/energy saving
  - Navigation & Tracking



Excerpted from ‘Sensor Network Survey’, ICU, 2005.

Transportation Applications





Excerpted from ‘Sensor Network Survey’, ICU, 2005.

Environment and Civil Engineering Applications

- **Environmental Monitoring**
  - Habitat Monitoring
  - Structural Monitoring

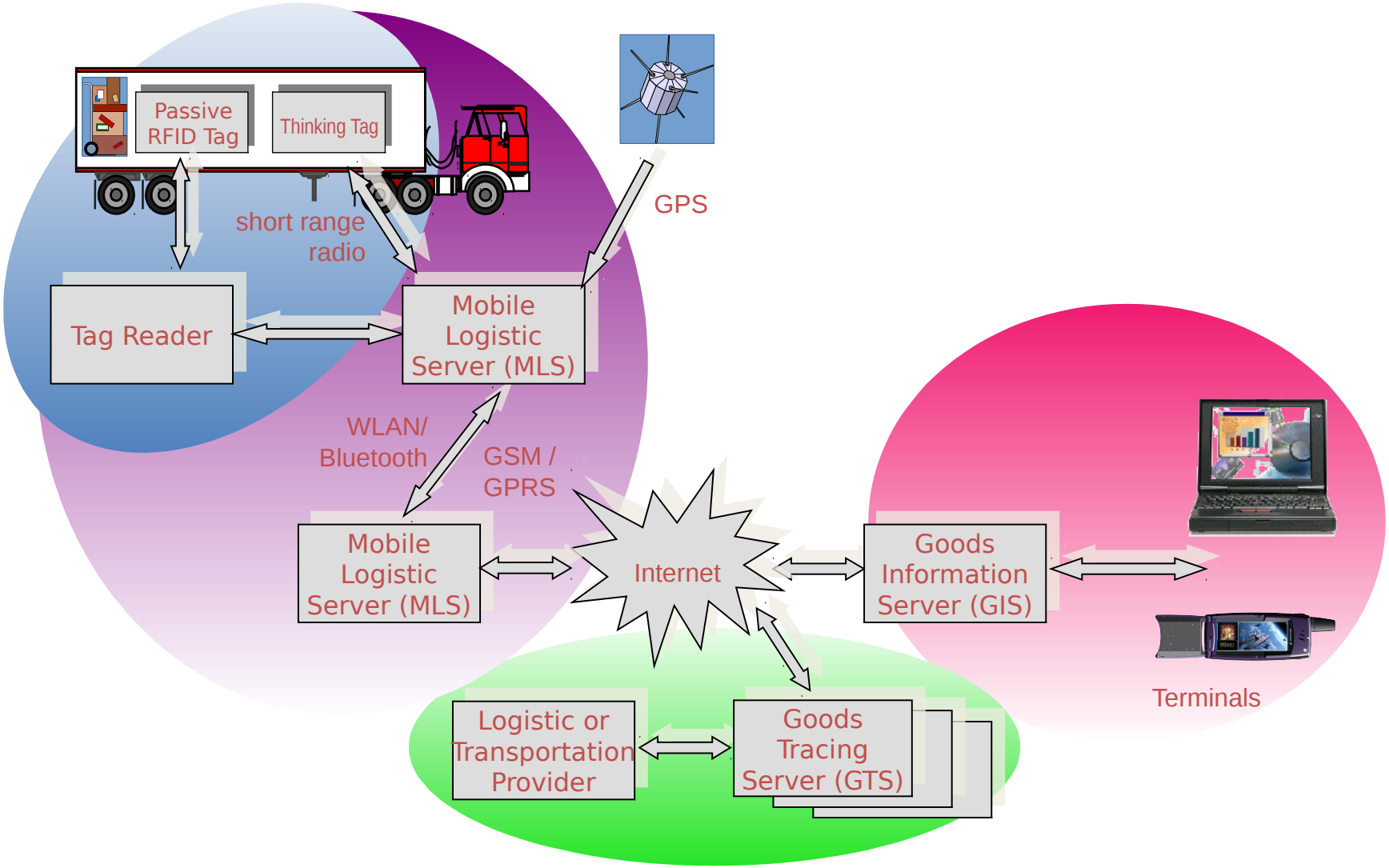
National Defense Applications

- Surveillance Platform

Excerpted from ‘Sensor Network Survey’, ICU, 2005.

Industrial Business Applications

- Supply Chain Management



Understanding our Evolution. What have we really evolved to?

- Internet of Thing
- Cloud Computing

Grand Challenges

- Large Scale Distributed processing requiring decentralization
- Long life-time requirement
- Reliability (no-repair failure model)
- Deployment
- Configuration and Network Management
- Programming the Internet of Things
- Learning by the Internet of Things
- Others.

Programming Pervasive Spaces

Cobol
ANSI Standard

Client/Server
TCP/IP

Synchronization
SyncML

?

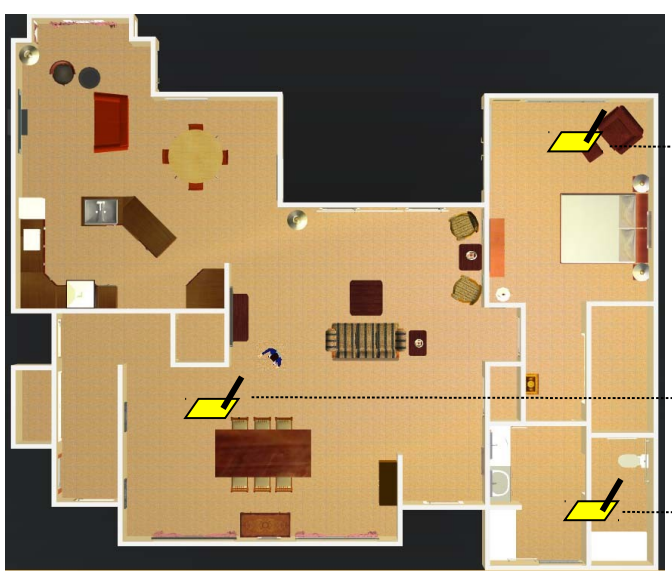
?

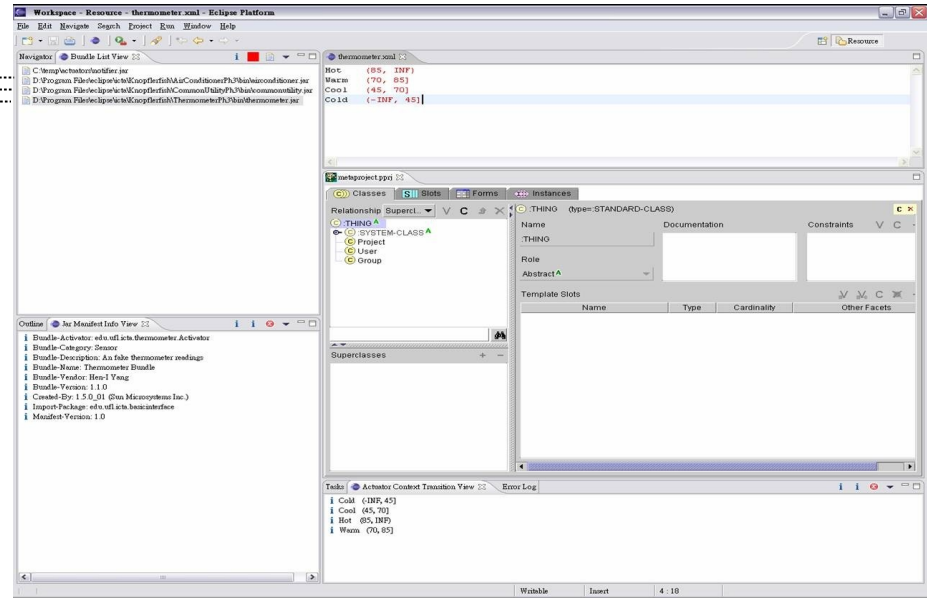
WSN in the Lab

WSN in the Field

WSN in the Field

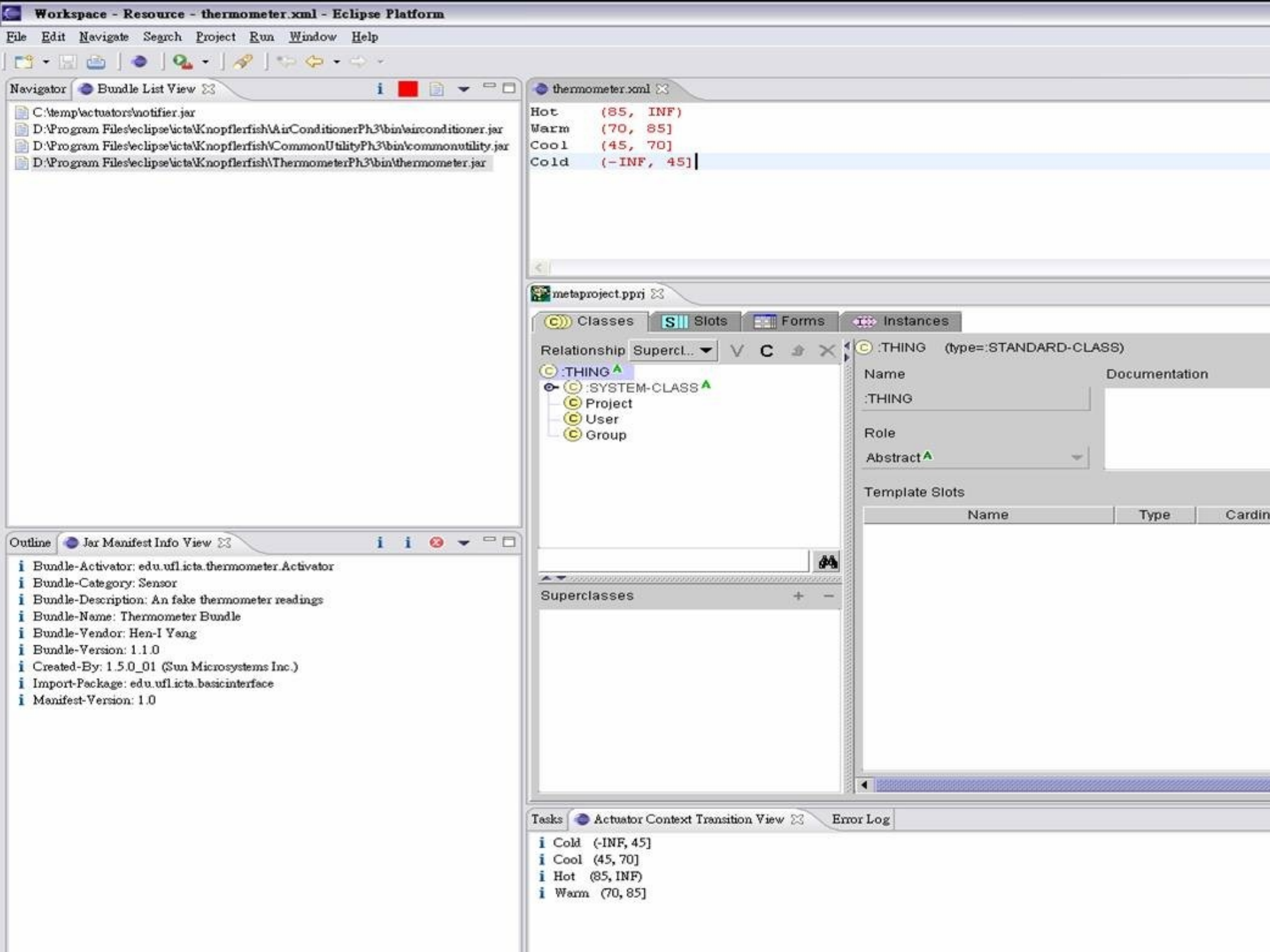
A view of Programmability in IoT: Plug & Play Sensor Network





3 Sensor Platforms Powered UP

3 OSGi Service Bundles appear in the IDE


Workspace - Resource - thermometer.xml - Eclipse Platform
File Edit Navigate Search Project Run Window Help
Navigator
Bundle List View
C:\temp\actuators\notifier.jar
D:\Program Files\eclipse\icta\Knopflerfish\AirConditionerPh3\bin\airconditioner.jar
D:\Program Files\eclipse\icta\Knopflerfish\CommonUtilityPh3\bin\commonutility.jar
D:\Program Files\eclipse\icta\Knopflerfish\ThermometerPh3\bin\thermometer.jar
thermometer.xml
Hot (85, INF)
Warm (70, 85]
Cool (45, 70]
Cold (-INF, 45]
metaproject.pprj
Classes
Slots
Forms
Instances
Relationship Supercl...
:THING
:SYSTEM-CLASS
Project
User
Group
:THING (type=:STANDARD-CLASS)
Name
:THING
Documentation
Role
Abstract
Template Slots
Name
Type
Cardin
Superclasses
Outline
Jar Manifest Info View
Bundle-Activator: edu.ufl.icta.thermometer.Activator
Bundle-Category: Sensor
Bundle-Description: An fake thermometer readings
Bundle-Name: Thermometer Bundle
Bundle-Vendor: Hen-I Yang
Bundle-Version: 1.1.0
Created-By: 1.5.0_01 (Sun Microsystems Inc.)
Import-Package: edu.ufl.icta.basicinterface
Manifest-Version: 1.0
Tasks
Actuator Context Transition View
Error Log
Cold (-INF, 45]
Cool (45, 70]
Hot (85, INF)
Warm (70, 85]

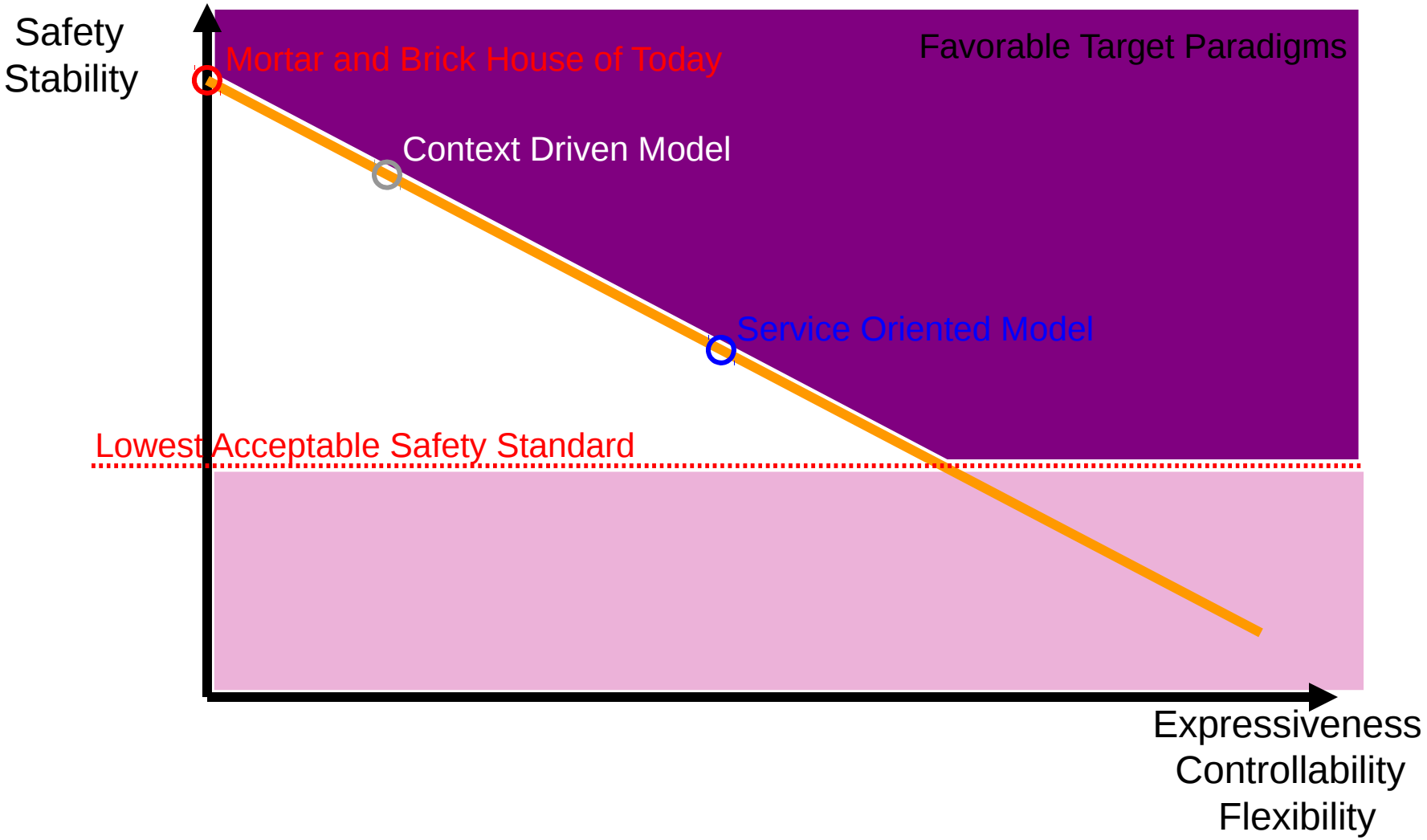
Programmable Things

- Plug & Play (self-integrative)
- Service-oriented architecture
  - Every sensor or actuator is converted into a software service, easily digested by programmers
    - Engineers Programmers (IT Industry)
- Programming & Application Development
  - Remote programming
  - Integrated Development Environments (IDE)

Context Aware Computing

- Define Contexts as special states of interest of the pervasive space
  - Example: day/night, hot/cold, just arrived, sleeping, etc.
- Use the states to guide the scope or set precondition for logic and action in the space
- Also, use states as taboos or “impermissible” contexts that should not be allowed to happen

Programming Models



Mobile and Pervasive Computing
CNT5517 - Section 2D32 & CIS4930 - Section 2D78
Lecture 8

Gator Tech Smart House

Dr. Sumi Helal
Computer & Information Science & Engineering Department
University of Florida, Gainesville, FL 32611
helal@cise.ufl.edu

Reading Materials

- A. Helal, W. Mann, H. Elzabadani, J. King, Y. Kaddourah and E. Jansen, "Gator Tech Smart House: A Programmable Pervasive Space", IEEE Computer magazine, March 2005, pp 64-74. (pdf)
- A. Helal, J. King, H. Zabadani and Y Kaddourah, "The Gator Tech Smart House: An Assistive Environment for Successful Aging," Book Chapter in "Advanced Intelligent Environments," H. Hagrass, Editor, Springer Verlag

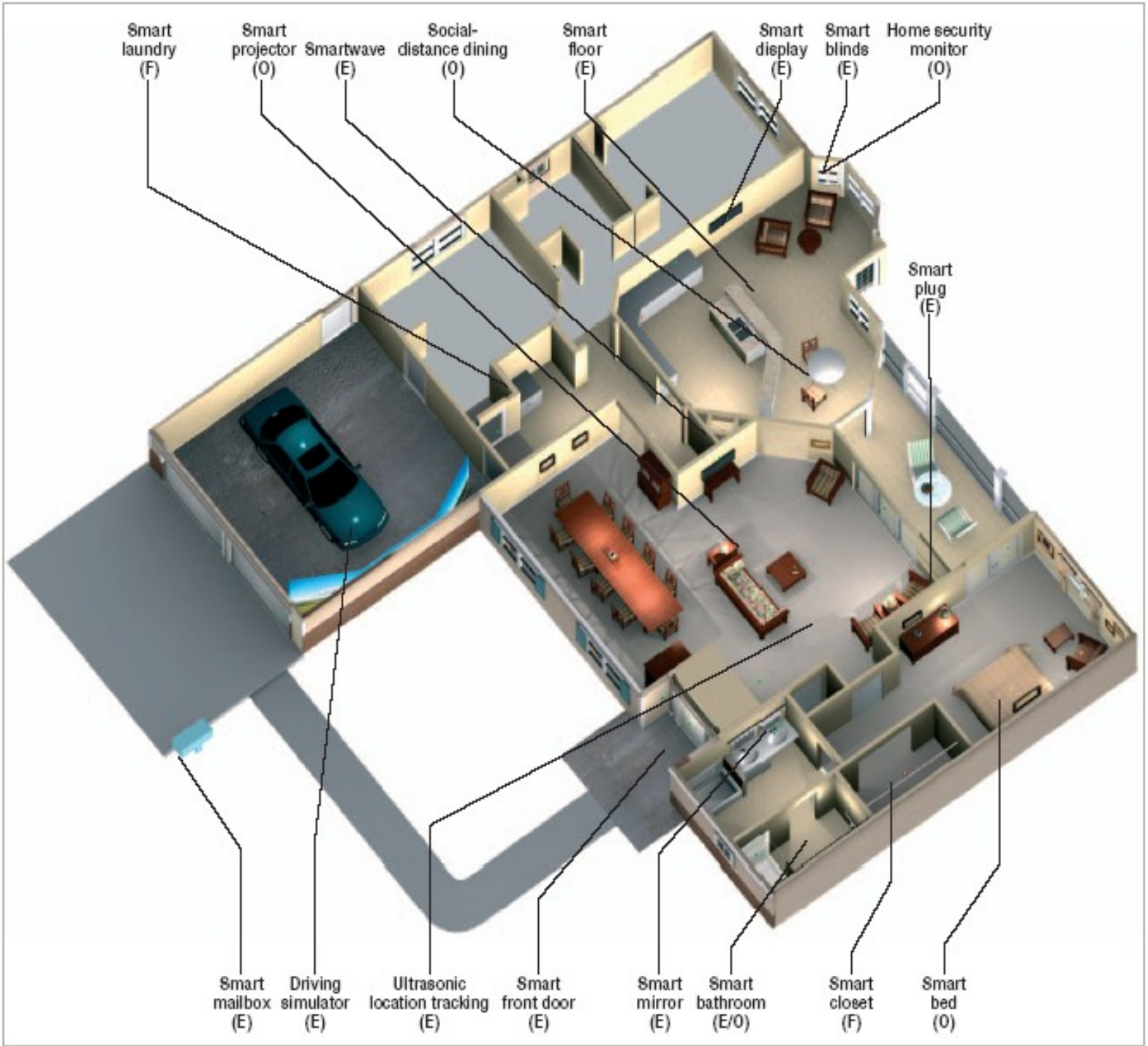
The Gator Tech Smart House

- What is it?
- What is its goal?
- What is it made of?

What is the GTSH?

- A 2500 sq ft single family house in the Oak Hammock Continuous Care Retirement Community, in Gainesville, Florida.
- A Pervasive Computing Space
- An experimental laboratory in which pervasive computing middleware and applications are innovated and validated by engineers as well as end users.

Floor Plan of the Gator Tech Smart House



What is the goal of the GTSH?

- Successful Aging
- Transform a home into an Assistive Environment
- Focus is on the elderly population and population of individuals with special needs.
- Performance metrics:
  - Quality of Life
  - Privacy Preservation
  - Cost
  - Scalability of Deployment

What is the GTSH made of?

- Dumb objects
- Sensors
- Actuators
- Devices
- Appliances
- Sensor Platforms

Sensors and Actuators

GTSH Sensors & Actuators List

- Pressure Sensors
  - Floor, Bed, Seats
- Contact/Proximity Sensors
  - Doors, Windows, Toilet, Microwave, Mailbox
- Microphone
  - Voice Control
- Cameras
  - Outdoor Security, Front Door
- Flow Meters
  - Sinks, Soap Dispenser
- Light
  - Blinds, TV-mode, Power Saving
- RFID Readers
  - Keyless Entry, Microwave, Smart Plugs
- Motion Sensors
  - Tracking
- Temperature Sensors
  - Climate Control, Oven
- Power Counters
  - Power Management, Appliance Use
- Moisture Sensors
  - Leak Detection
- Barcode Reader
  - Medicine Reminder

- Infrared Sensors
  - Resource Monitoring
- Push-button Sensors
  - Doorbell, Light Switches
- Servos
  - Blinds, Front Door Deadbolt
- Door Opener
  - Keyless Entry, Voice Control
- TVs
  - Entertainment, Notification
- Speakers
  - Entertainment, Notification
- Microwave
  - Cooking Assistant
- Camera PTZ
  - Security
- X10
  - Appliance Control

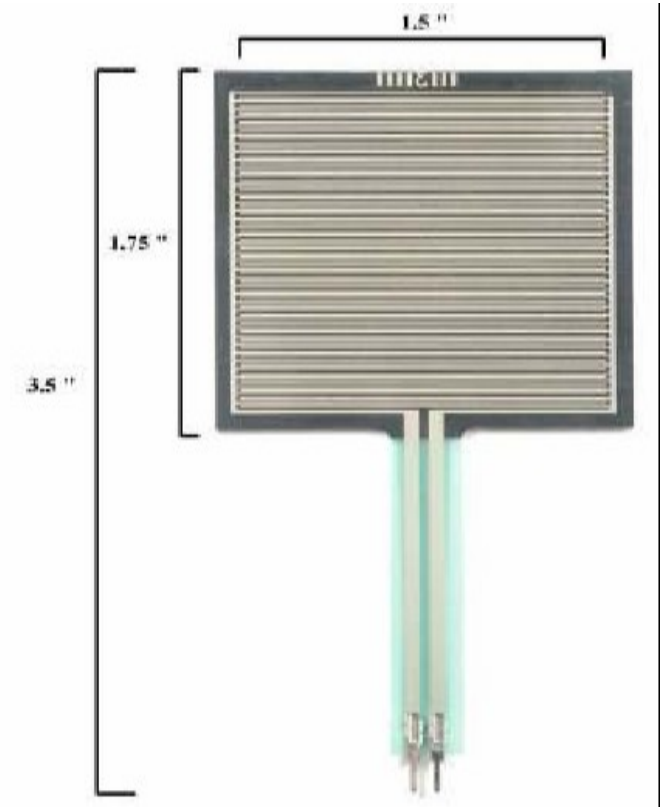
Simple Sensors

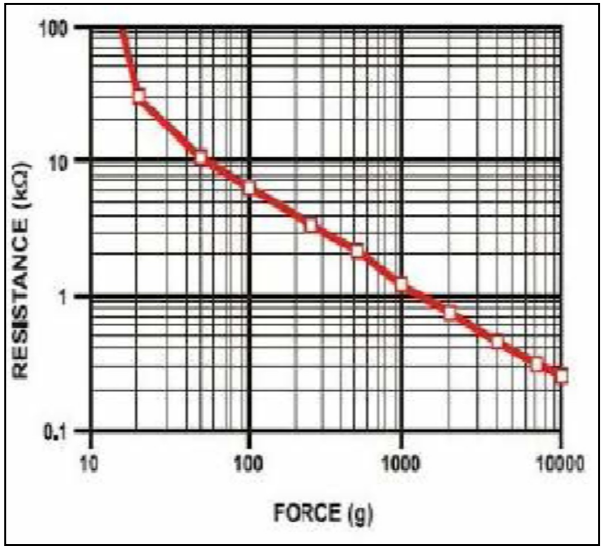
- **ANALOG**
- Can provide a range of values
- Generally requires 3 pins (wires) to use:
  - Power / Reference Voltage
  - Ground
  - Input
- Requires ADC

- **DIGITIAL**
- Digital output
- No ADC required

Pressure / Force Sensitive Resistors (FSR)

- Polymer Thick Film (PTF) Device: Piezo-electric (actually Piezo-resistive)
- As force increases, resistance decreases

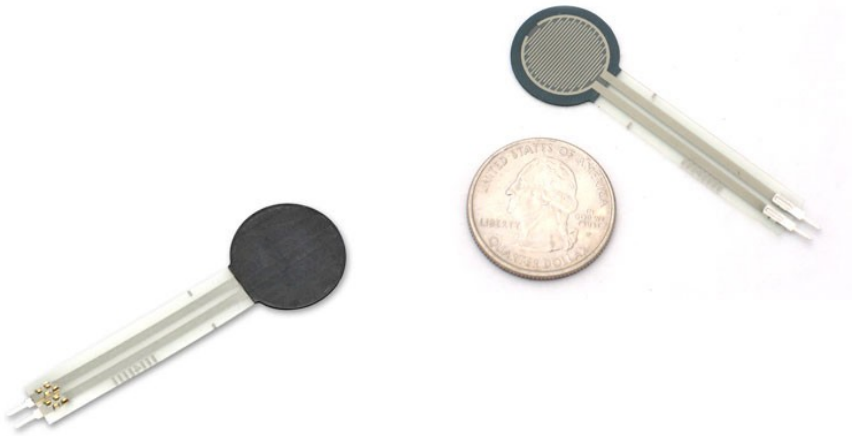


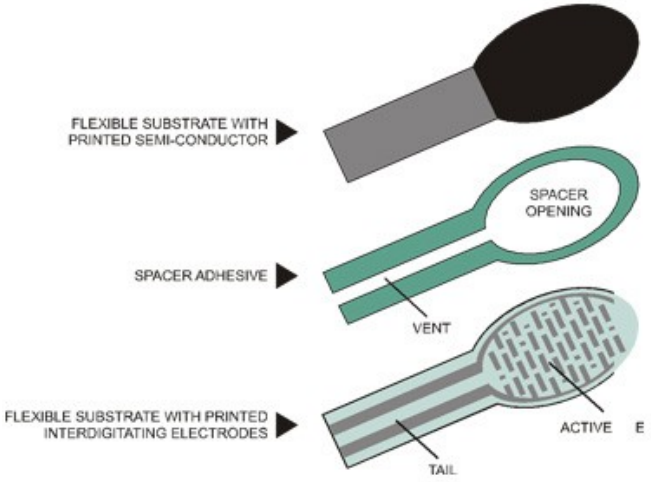


FSR Salient Characteristics

- Force Sensitivity:
  <100g to >10kg
- Pressure Range:
  <1.5 psi to >150 psi*
- Break Force:
  20g to 100g
- Device Rise Time:
  1-2 ms
- Temperature Range:
  -30^{o}C to +70^{o}C

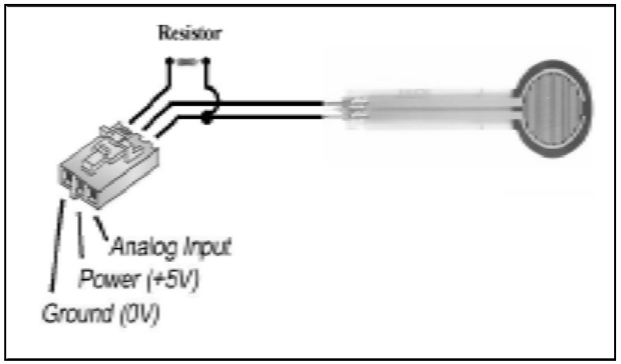
* Pound per square inch





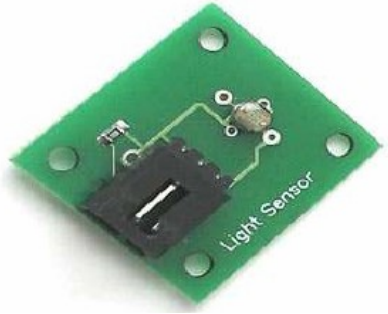
Conditioning a Sensor for a Specific Application

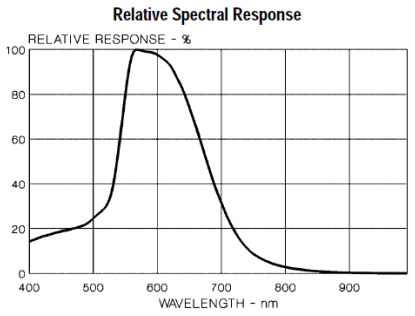
- Let us talk about FSR
- Sensor has two pins
- Need to add resistor (requires 3 pins) to control sensitivity of sensor
- Higher resistor value means more sensitivity:
  - Can detect small changes in force
  - Smaller changes in force result in greater changes in resistance
  - Also means a smaller force will max out the reading
  - Measuring heavier forces requires smaller resistors

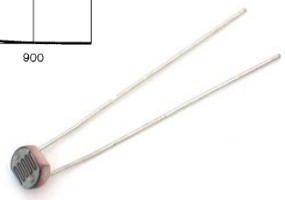


Light Sensor

- Photoresistor*: Resistance decreases with increasing light*
- High-resistance semiconductor
- High-frequency light = photons absorbed, give electrons energy to jump into conduction band
- Free electrons = more conductive = less resistance
- Cadmium sulphide, 2 M in darkness, 500 in bright light



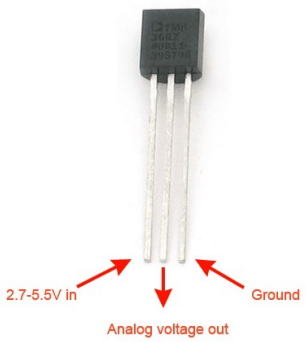




Temperature Sensors

- Thermistor
  - Resistance changes with temperature
  - NTC: Negative Temperature Coefficient,
  - PTC: Positive Temperature Coefficient, “switches” to high resistance at critical temperature
  - Good for measuring small temperature changes with high accuracy over small range

- Thermocouple
  - When conductor subject to thermal gradient, generates voltage
  - Voltage magnitude depends on conductor material
  - Use dissimilar conductor to complete circuit, will have different generated voltage, measure difference
  - Voltage difference grows with temperature
  - Very small voltages! 1-70 uV per ^{o}C!
  - Rugged, works over wide temperature range, but require specialized equipment, lots of industrial applications



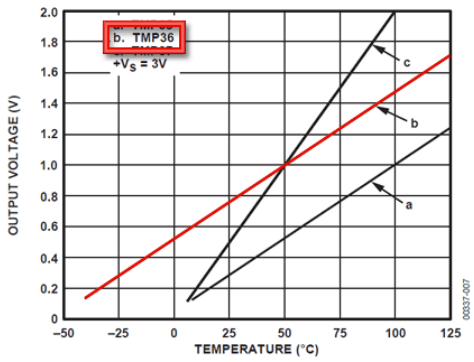
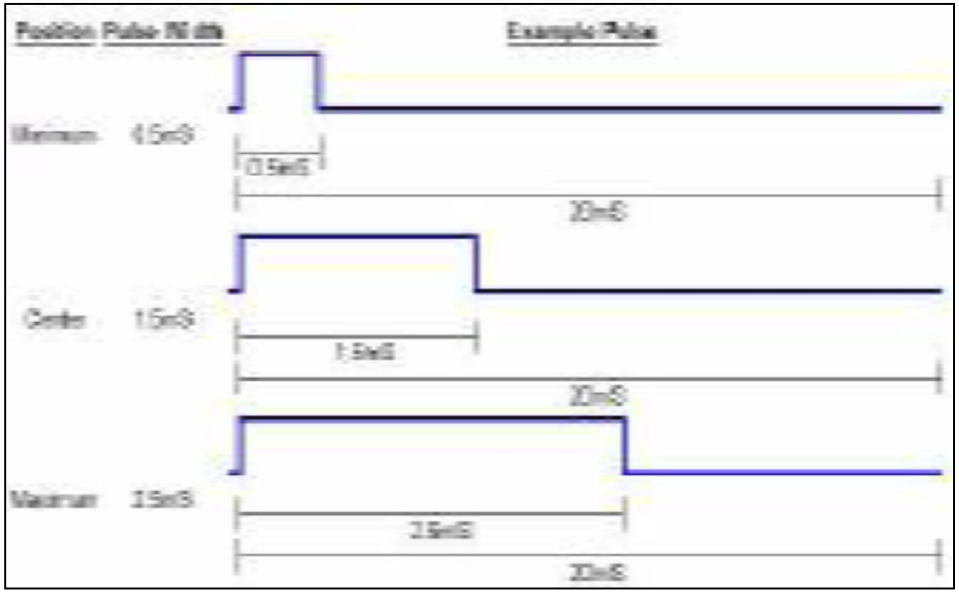


Figure 6. Output Voltage vs. Temperature

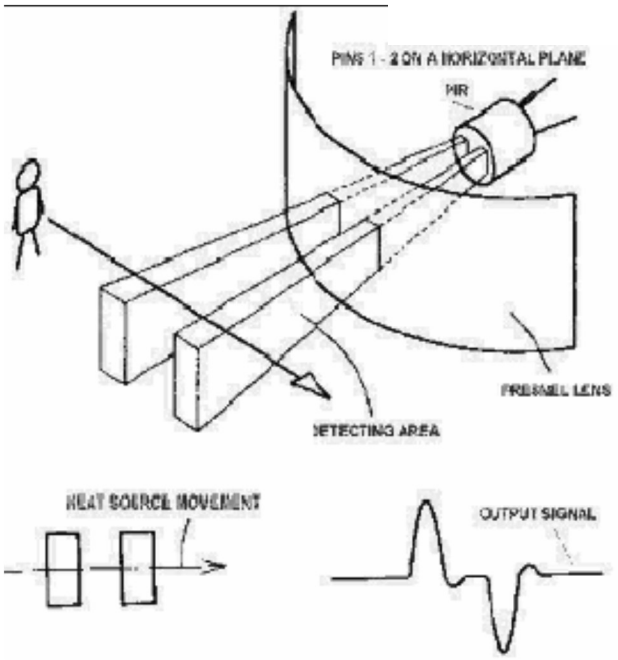
Servos

- Positionable (step) motors
- 3 Pins: Power,Ground, Command
- Usually digital, not analog
  - No ADC
  - Less power
  - More noise-proof, longer runs
- Set speed, torque



Motion Sensor (Passive Infrared or PIR sensors)

- Basically pair of IR light sensors and filter that blocks non-IR light
- Check for input on one, then the other



Temperature/Humidity Sensor

- Pin 1: Temperature
  0 to Vsupply =
  -30 to +100^{oC}
- Pin 2: Vsupply (2 to 5.5V)
- Pin 3: Humidity

  0 to Vsupply =

  0 to 100%
- Pin 4: Ground
- RH Accuracy: +/-2.0%
- RH Response: 25 s
- Temp Accuracy: +/-0.40oC
- Temp Response: 50 s
- Stabilization: 5 min

Private-Door Duo

- A latch strike mechanism
- Digital Device, 12V
- Smart Actuators, integrated controller
- Switches state (open/close) on voltage pulse

Network Camera

- Combination Sensor & Actuator
- Smart Device, integrated microcontroller, runs web server
- Image stream, PTZ commands over HTTP
- CCDs and Servos

Devices

Blood Pressure Measuring Device

Knowing your devices

- Thermistor Example
  - Linear function? $\Delta R = k\Delta T$?
  - No. Steinhart-Hart eqn, 3rd order approximation:

  $1/T = a + b \ln R + c \ln^3 R$

  a, b, c are Steinhart-Hart parameters, can vary

  across devices, typical values:

  1.40 x 10-3, 2.37 x 10-4, 9.90 x 10-8

- Self-heating effects! Powering sensor generates heat that is detected.

Knowing your devices

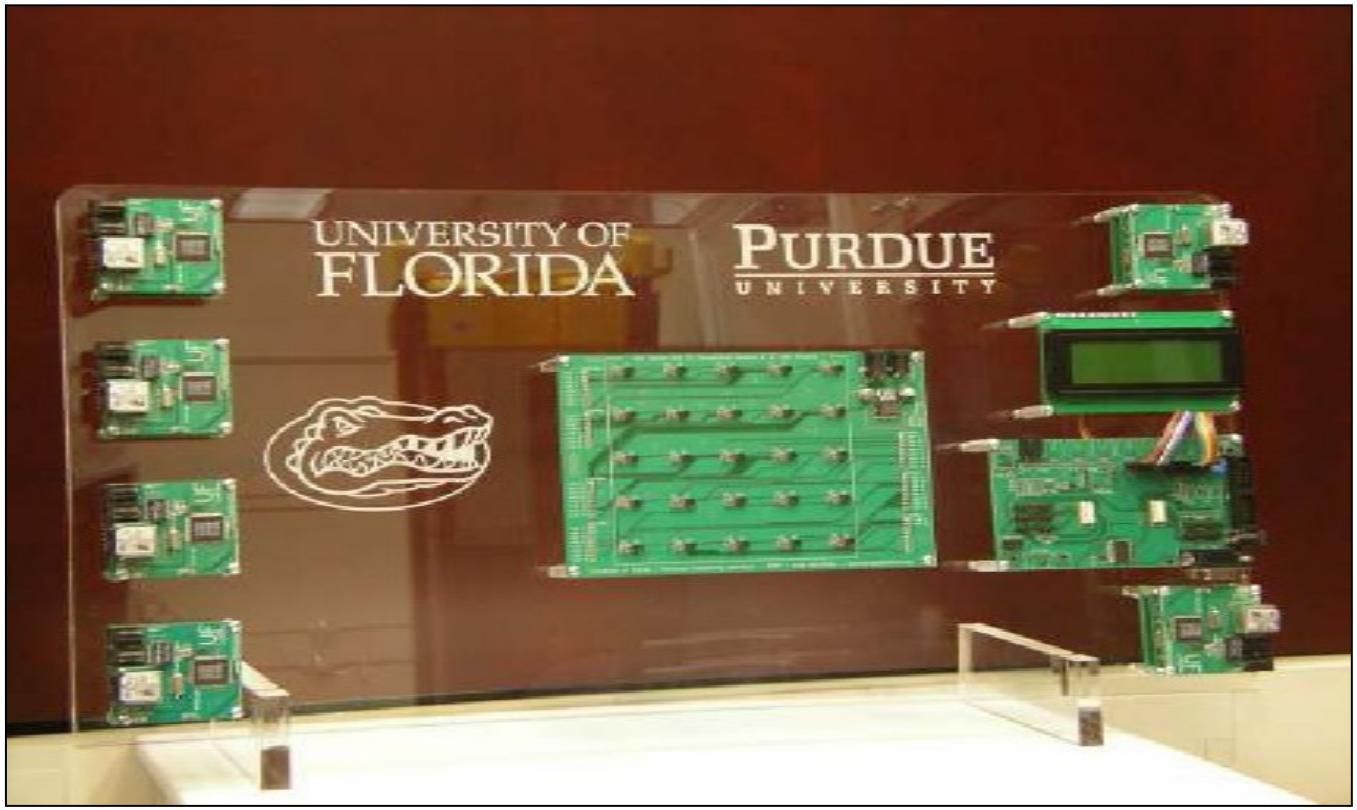
- Servo Example
  - Rotation speed constant? Same PWM (control) will always put into the same position?
  - No. Load on servo affects both.
  - May be continuously driven if carrying a heavy load.

Connecting devices into a network

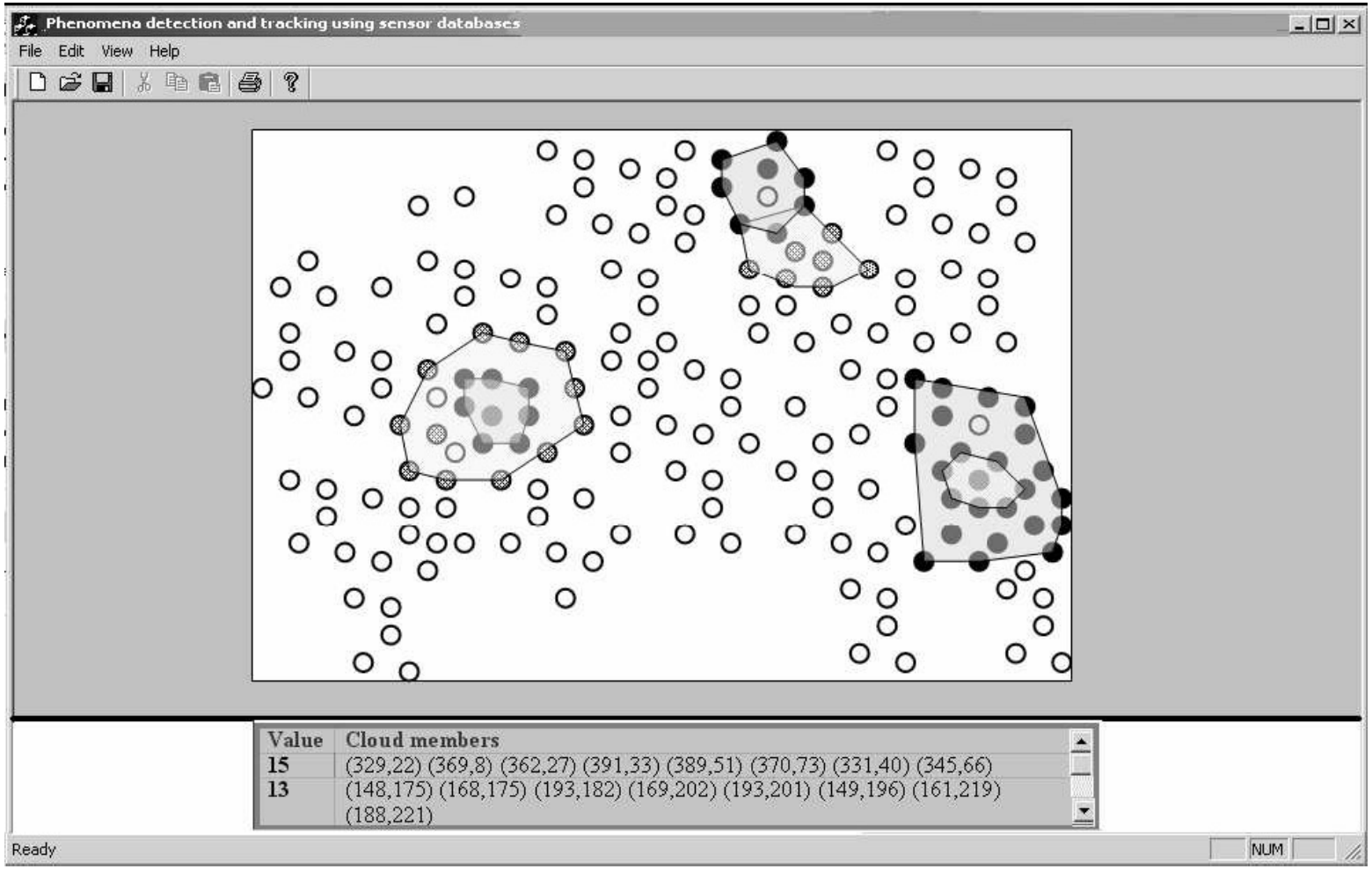
- How to bring devices together to do something useful?
- Analog vs. Digital Issue
- Digital devices could be connected directly to the computer (serial, parallel port) if voltages are compatible
- Analog devices need ADC
- Smart devices may be too smart for their own good, setup to use some specific network tech that nothing else is using
- OR ….
- **Use Sensor (and Actuator) Platforms**

Example: Network of 25 sensors
The Purdue Nile-PDT Demo

- Using 6 sensor platforms
- Using a debug board and an LCD panel

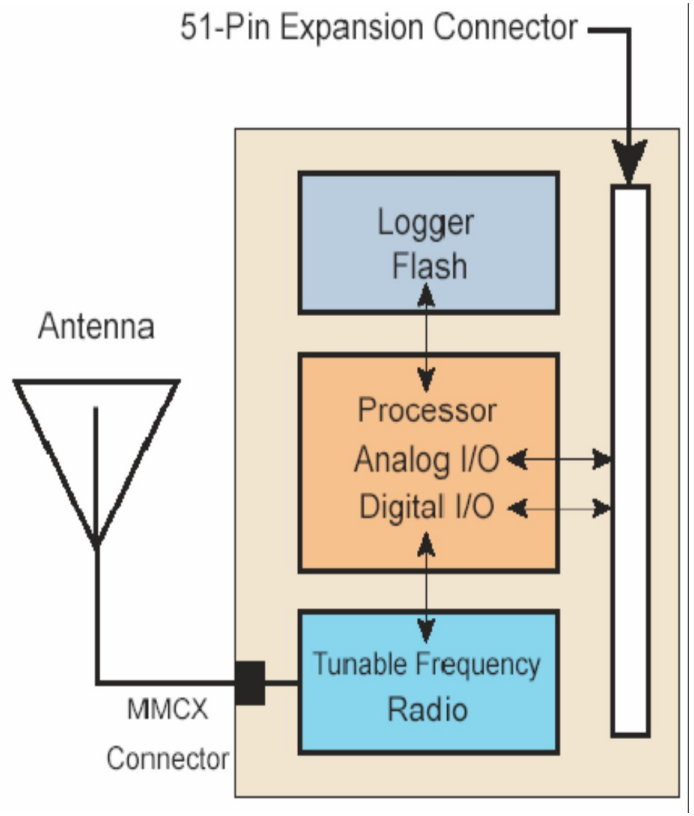


Example: Purdue NILE-PDT



Sensor Platforms

Mica2 Family - 2002



- ADC
- Digital
- I/O
- I2C*
- SPI
- UART

Serial Peripheral Interface Bus

Universal asynchronous receiver/transmitter

* Inter-Integrated Circuit Bus (by Phillips).

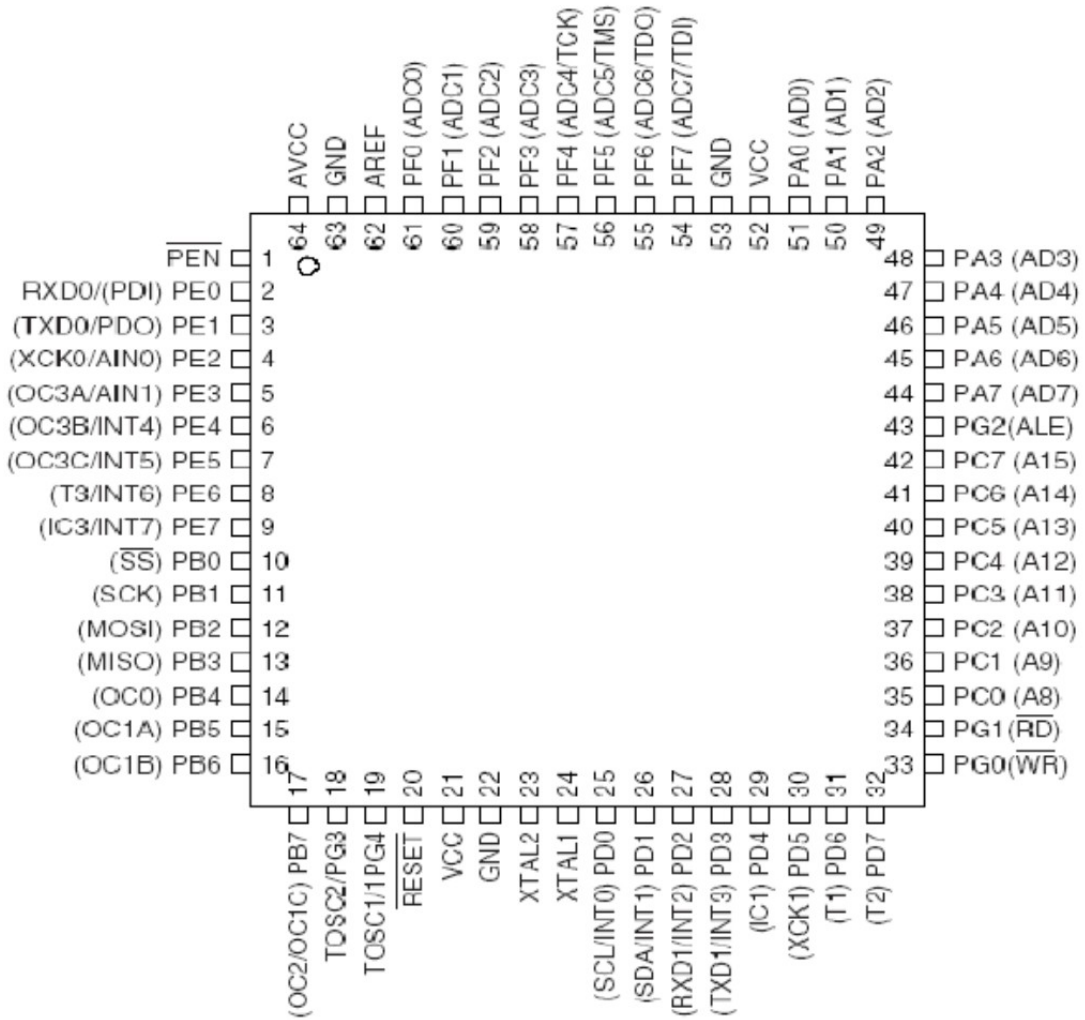
Mica2 Hardware

- Processor
  - Atmel ATmega128L
  - 128K Program Flash
  - 4K SRAM
  - 4K EEPROM
  - 10-bit ADC
  - UART, DIO, I2C, SPI
  - 8 mA Current Draw
- Network
  - Chipcon CC1000
  - Various frequencies in ISM band
  - Proprietary RF protocol
  - 38.4 Kbaud
  - 1000 ft outdoor range
  - 25 mA max transmit
  - 8 mA Receive

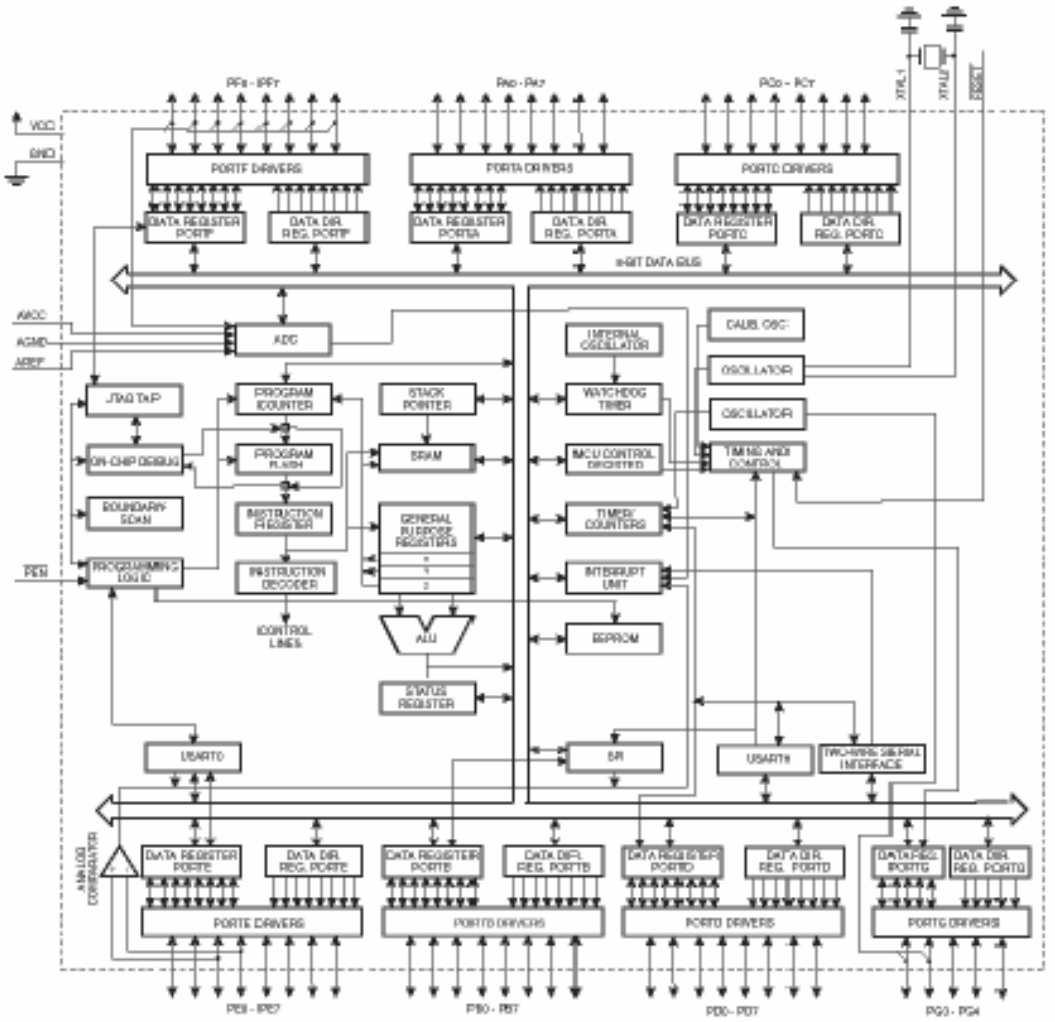
Atmel ATmega128L

- 8-bit microcontroller, Harvard architecture, ‘Enhanced’ RISC
- 2 – 16 MHz
- 133 instructions, most single-clock
- 32 x 8 registers
- 128K program flash
- 4K EEPROM
- 4K SRAM
- Up to 64K external memory
- 2 x 8-bit, 2 x 16-bit timer/counters
- 2 x 8-bit PWM channels, 6 programmable 2-16 bit PWM
- 8-channel, 10-bit ADC
- 2 x Serial UARTs
- Watchdog timer
- 6 sleep modes
- SPI, I^{2C}
- 53 programmable I/O lines

ATmega128L Pinout



ATmega128L Block Diagram



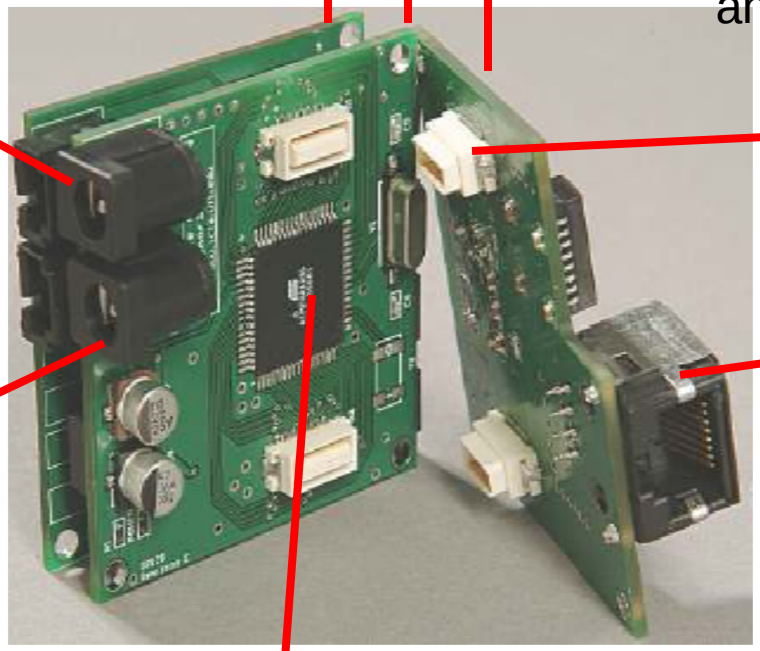
Telos - 2004

- Similar to Mica2 family
- Uses TI MSP430 microcontroller
  - 8 MHz
  - 48K program flash,
  - 10k SRAM
  - 3 mA current draw
- Chipcon CC2420 radio
  - 125m outdoor range
  - IEEE 802.15.4 PHY/MAC

Atlas - 2005

Layered Design
For flexible configuration of processing, power, communication, and sensor/actuator needs.

Power
Wired power option for use with indoor applications.



Quick Connect
For easy and reliable stacking.

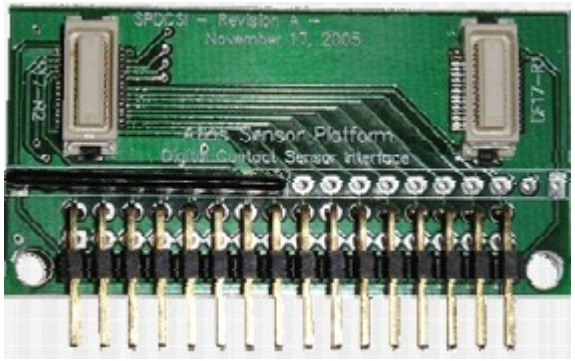
Networking
Swappable communication layers to support different mediums. Ethernet, WiFi, ZigBee, USB.

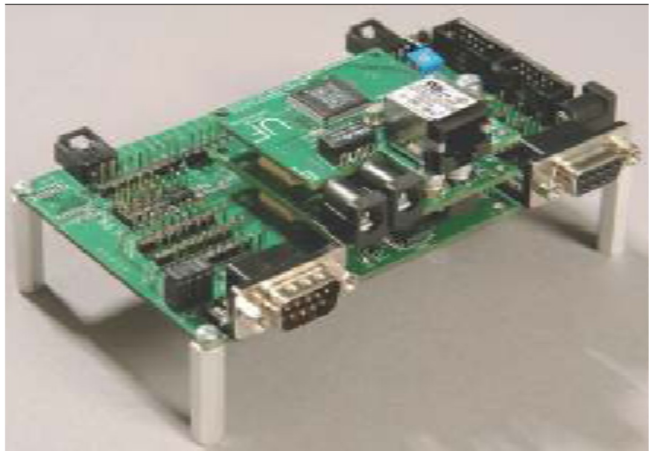
More Power
Daisy-chain sensor platforms to create large networks without tying up all outlets.

Processor
ATmega128 provides low-cost and low-power processing. Runs OS that monitors sensor connections and communicates with server. Internal storage for sensor/actuator OSGi bundles and da accumulation for on-node processing.

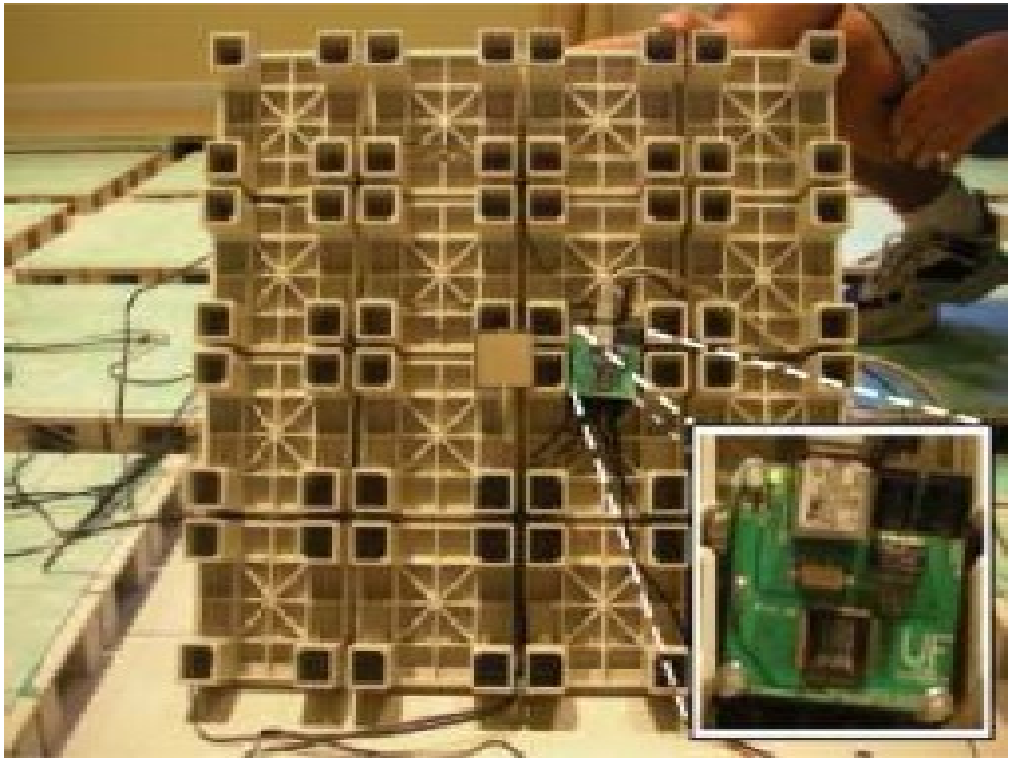
Atlas Communication Layers

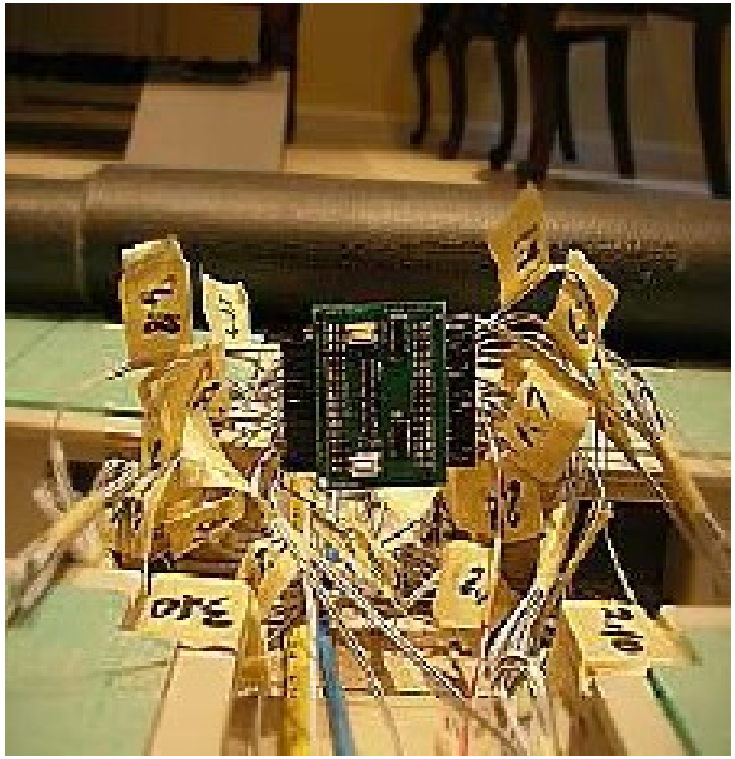
Atlas Interface Layers





GTSH Smart Floor





Mobile and Pervasive Computing

CNT5517 - Section 2D32 & CIS4930 - Section 2D78

Lecture 9

The Atlas Sensor Platform & Middleware Architecture

Dr. Sumi Helal

Computer & Information Science & Engineering Department

University of Florida, Gainesville, FL 32611

helal@cise.ufl.edu

Reading Materials

- J. King, R. Bose, H. Yang, S. Pickles and A. Helal, “Atlas – A Service-Oriented Sensor Platform,” Proceedings of the first IEEE International Workshop on Practical Issues in Building Sensor Network Applications (SenseApp 2006). Tampa, Florida, November 2006. (pdf)
- R. Bose, J. King, H. El-zabadani, S. Pickles, and A. Helal, "Building Plug-and-Play Smart Homes Using the Atlas Platform," Proceedings of the 4th International Conference on Smart Homes and Health Telematic (ICOST), Belfast, the Northern Islands, June 2006. (pdf)
- The Device Description Language (DDL) Specifications: http://www.icta.ufl.edu/atlas/ddl/

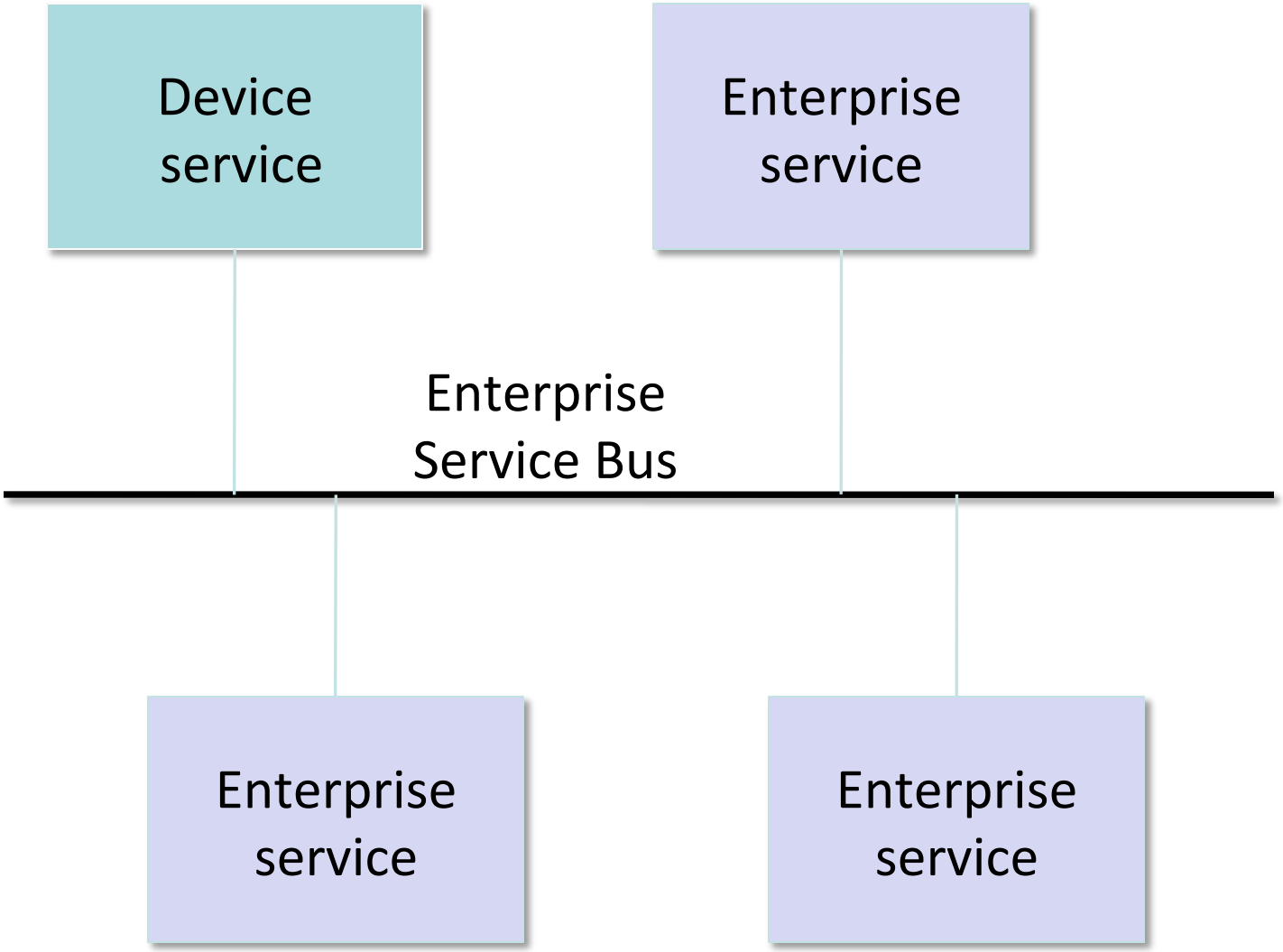
Goals of Atlas

- Enable continuous self-integration of sensors, actuators and devices
- Allow for programming and configuring a deployed space
- Decouple deployment (instrumentation) from programming & application developments
- Define and support a manageable full life cycle for pervasive spaces.
- Contribute to standard processes promoting openness and interoperability across various pervasive technologies.

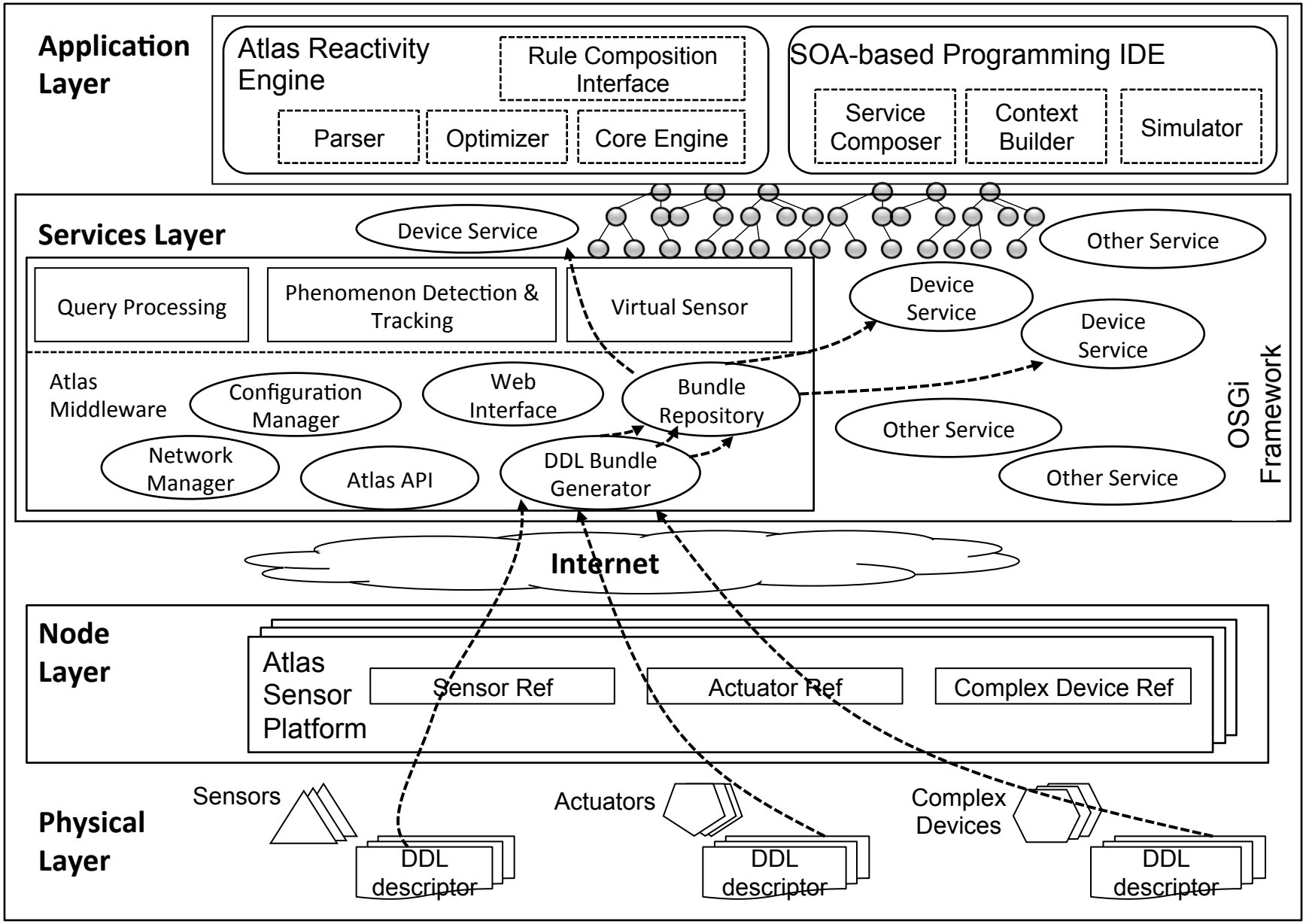
Atlas Architecture

- Service Oriented Architecture (SOA), in which each sensor, actuator or device is represented as a software service
- Based on the Open Services Gateway Initiative (OSGi), which is a Java based service framework
- Plug and Play, in which sensors or devices are discovered and included in the framework on power up. Similarly, devices are unplugged as they are turned off or fail
- Accommodates both pinhead sensors and sophisticated devices
- Contributes to a reference implementation of the SODA standard (Service Oriented Device Architecture)

Enterprise View of the Atlas Architecture

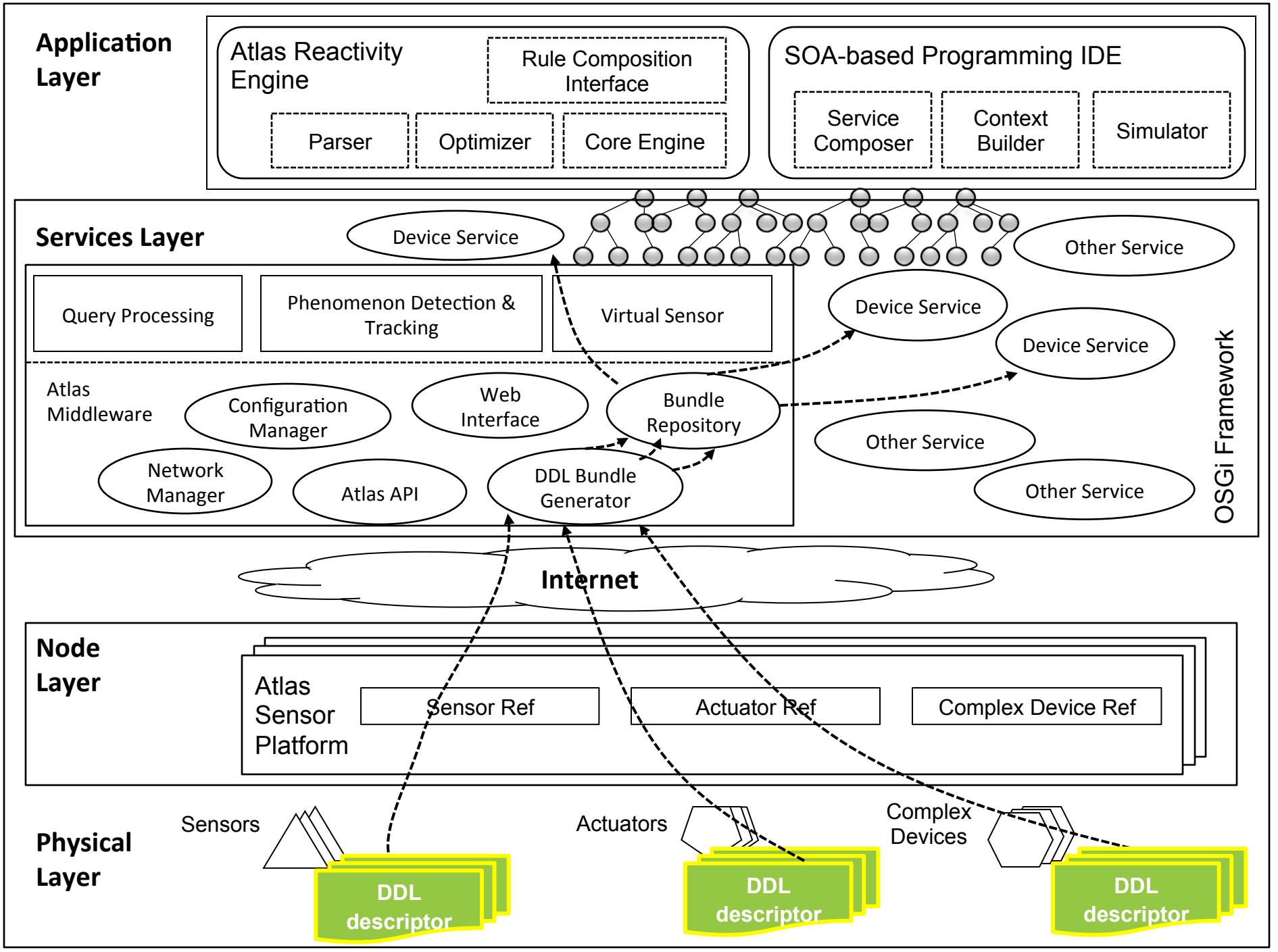


The ATLAS Architecture



Atlas in a Nut Shell

- Represents each of the devices (sensors and actuators) connected to it as software services
- New devices automatically appear as services inside service framework upon power-up.
- Application Developers do not require hardware knowledge or need to learn embedded systems programming to integrate sensors and actuators into their applications.
- Sensors and actuators can be controlled through high-level software methods
- Uses the OSGi framework to represent devices as service bundles


Application Layer
Atlas Reactivity Engine
Rule Composition Interface
Parser
Optimizer
Core Engine
SOA-based Programming IDE
Service Composer
Context Builder
Simulator
Services Layer
Device Service
Other Service
Query Processing
Phenomenon Detection & Tracking
Virtual Sensor
Device Service
Device Service
Atlas Middleware
Configuration Manager
Web Interface
Bundle Repository
Other Service
Network Manager
Atlas API
DDL Bundle Generator
Other Service
OSGi Framework
Internet
Node Layer
Atlas Sensor Platform
Sensor Ref
Actuator Ref
Complex Device Ref
Physical Layer
Sensors
Actuators
Complex Devices
DDL descriptor
DDL descriptor
DDL descriptor

DDL: Device Description Language

- An open standard to allow device and sensor vendors to introduce self-integrating products into the market place.
- DDL - A descriptive language that describes a broad range of devices.
- A reference implementation of DDL in ATLAS

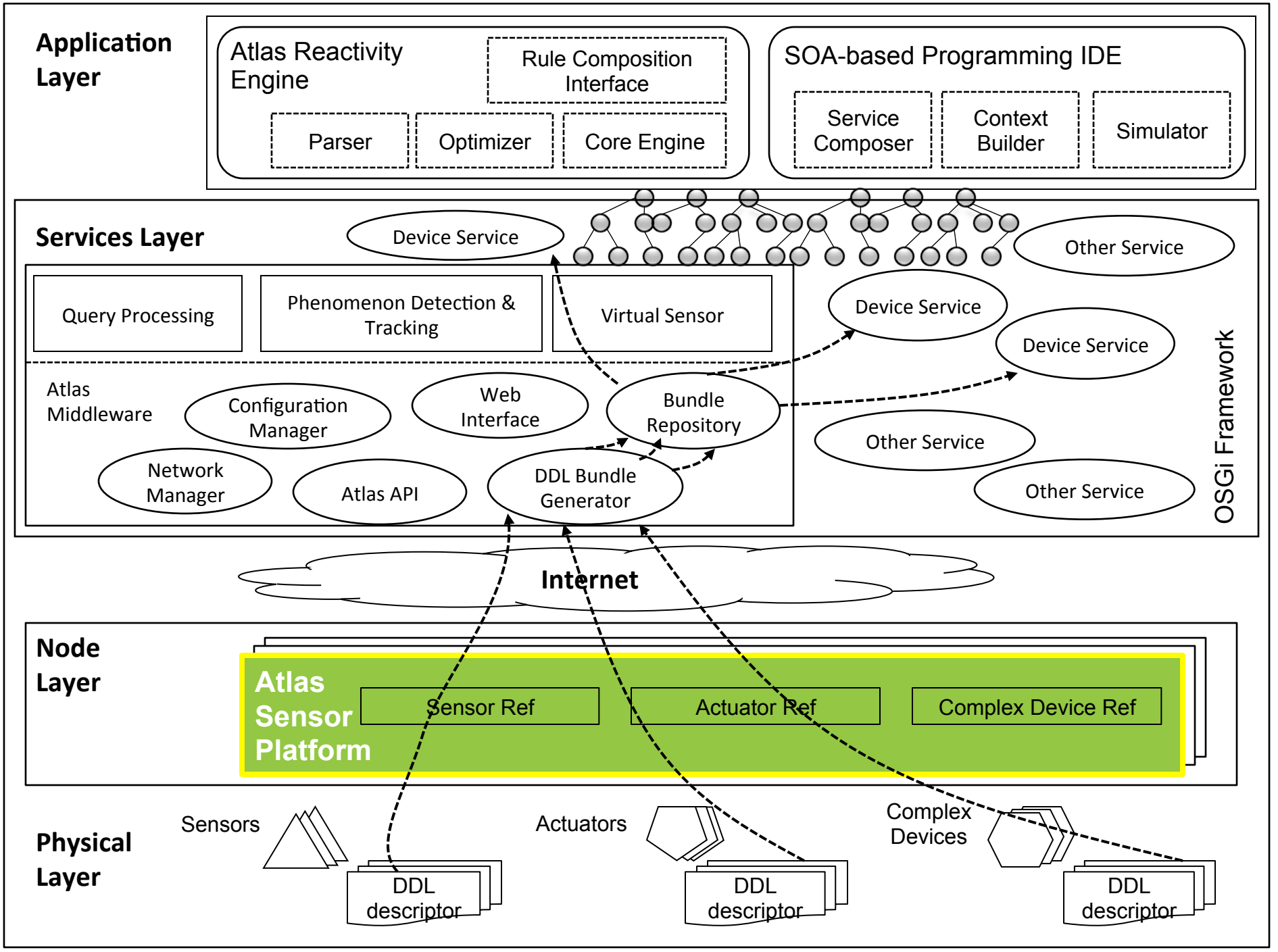
AN EXAMPLE: TMP36 ANALOG TEMPERATURE SENSOR

- Each DDL descriptor file describes a single type of device
- It contains:
  - Information for service registration and discovery
    - e.g., device name, model, function description, etc.
  - Description of device operations
    - each operation is a collection of *input/processing/output* function chains
    - the low-level communication between a device and its service are represented as **'Signals'**
    - the high level semantics of signals are **'Readings'**

```
<Sensor>
<Description>…</Description>
 <Interface>
  <Signal id="ADC1">…</Signal>
  <Reading id="Temp1">
   <Type>Physical</Type>
   <Measurement>Temperature </
Measurement>
   <Unit>Centigrade</Unit>
   <Computation>
  <Type>Formula</Type>
  <Expression> Temp1 =
(((ADC1/1023) * 3.3)-0.5)*
(1000/10)</Expression>
   </Computation>
  </Reading>
 </Interface>
</Sensor>
```

STANDARD SPECIFICATION

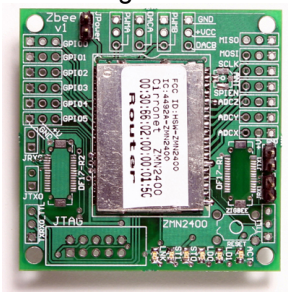
- The DDL language specification and its software are available online at
  - http://www.icta.ufl.edu/atlas/ddl/


Application Layer
Atlas Reactivity Engine
Rule Composition Interface
Parser
Optimizer
Core Engine
SOA-based Programming IDE
Service Composer
Context Builder
Simulator
Services Layer
Device Service
Other Service
Query Processing
Phenomenon Detection & Tracking
Virtual Sensor
Device Service
Device Service
Atlas Middleware
Configuration Manager
Web Interface
Bundle Repository
Other Service
Network Manager
Atlas API
DDL Bundle Generator
Other Service
OSGi Framework
Internet
Node Layer
Atlas Sensor Platform
Sensor Ref
Actuator Ref
Complex Device Ref
Physical Layer
Sensors
Actuators
Complex Devices
DDL descriptor
DDL descriptor
DDL descriptor

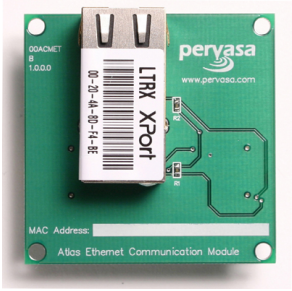
ATLAS Sensor Platform

Atlas Communication Layer

ZigBee

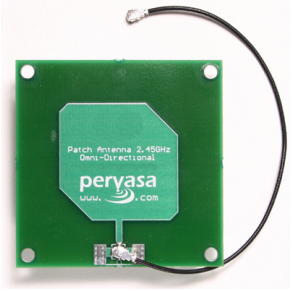


Ethernet



Wi-Fi

Antenna

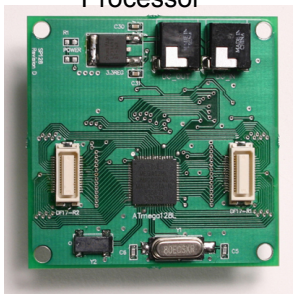


best of sensors expo 2007

peryasa

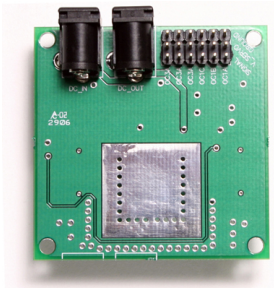
Atlas Processing Layer

Processor



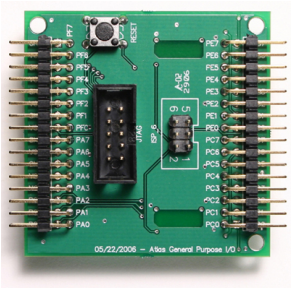
Atlas Device Interface Layer

Servo

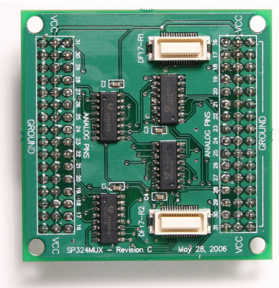


8 Analog

GPIO



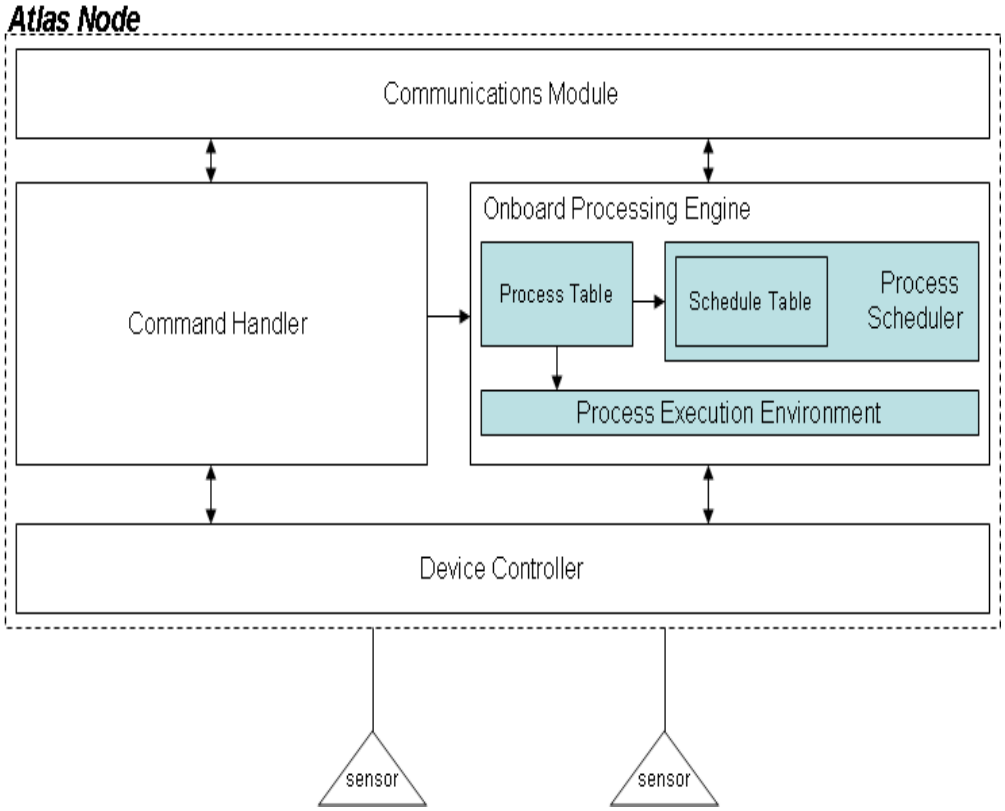
32 Analog



16 Digital

ATLAS Firmware Components

- Device Controller
- Communications Module
- Command Handler
- Onboard Processing Engine



Plug-n-Play Step 1

Step 1 connecting the sensors & actuators …

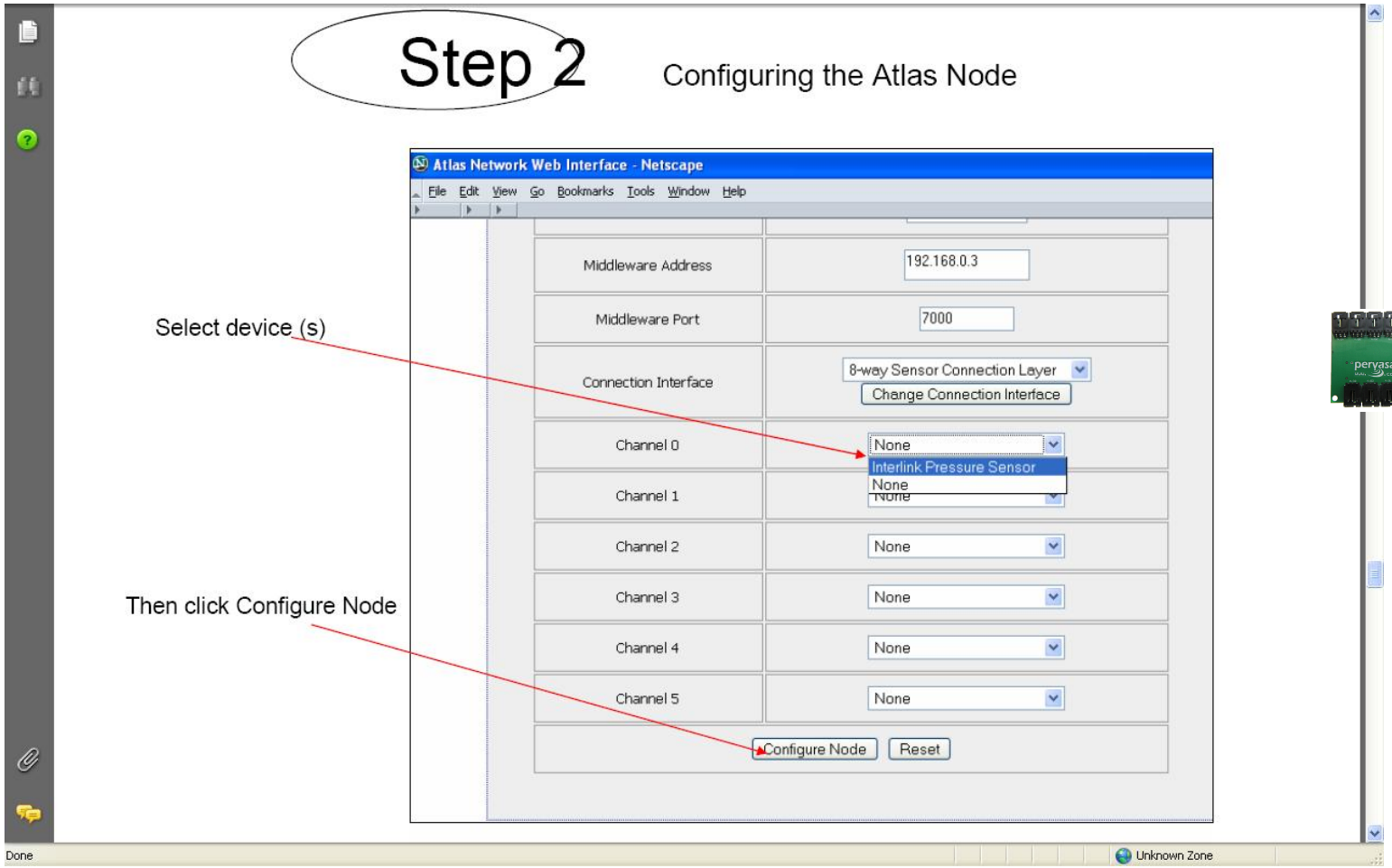
sensor

Atlas Node

Done

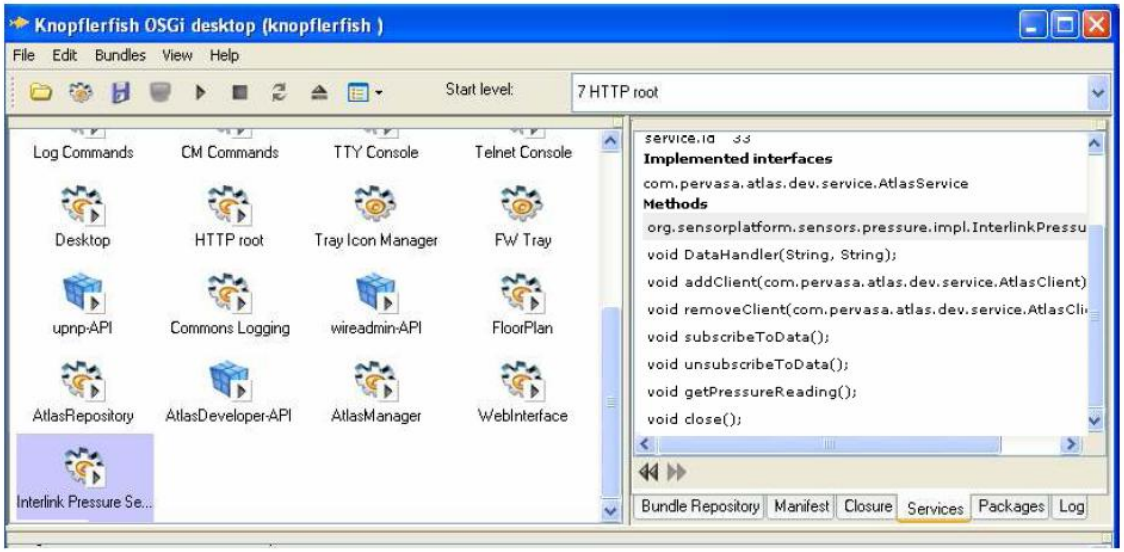
Unknown Zone

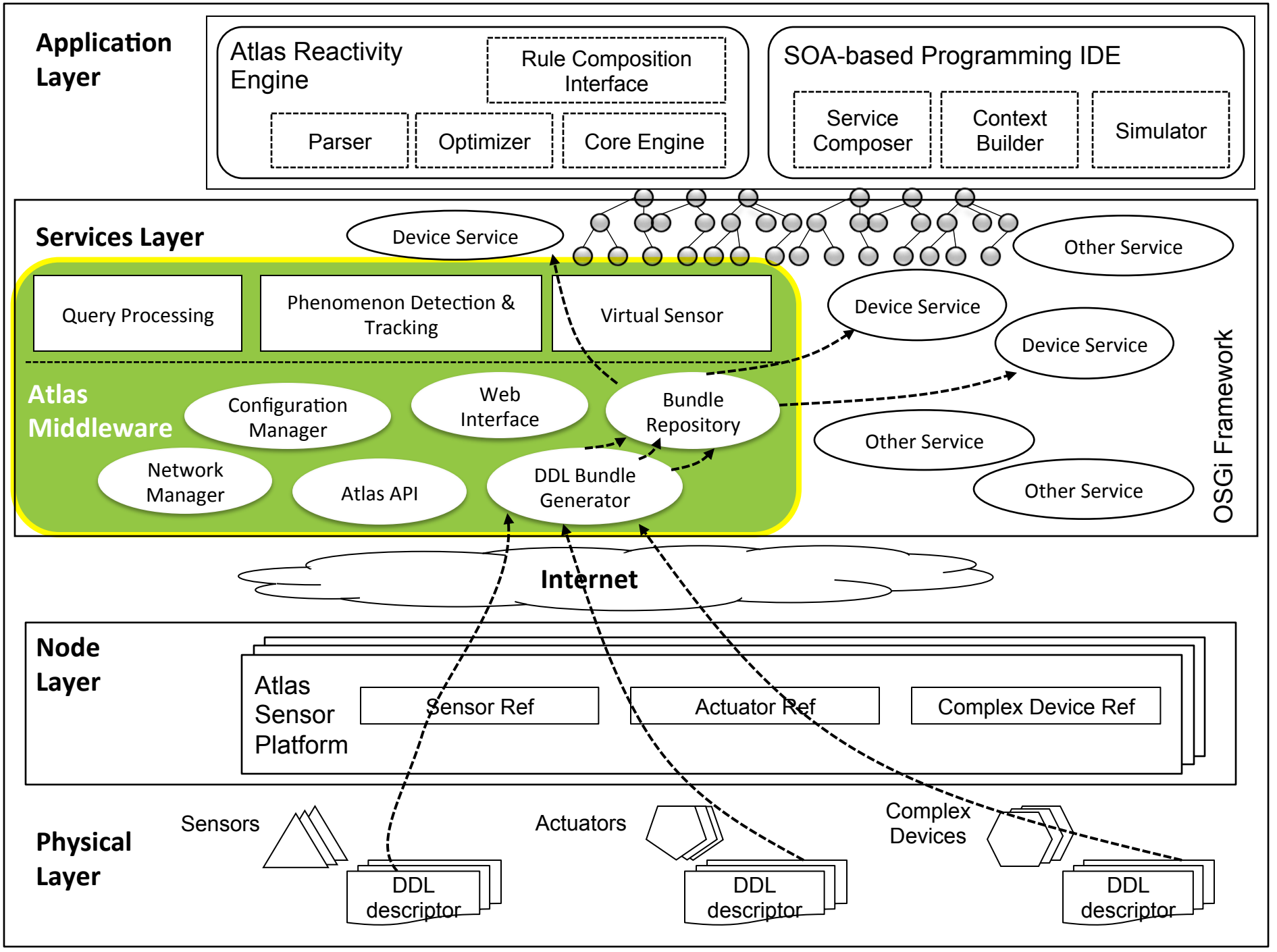
Plug-n-Play Step 2



Plug-n-Play Step 3

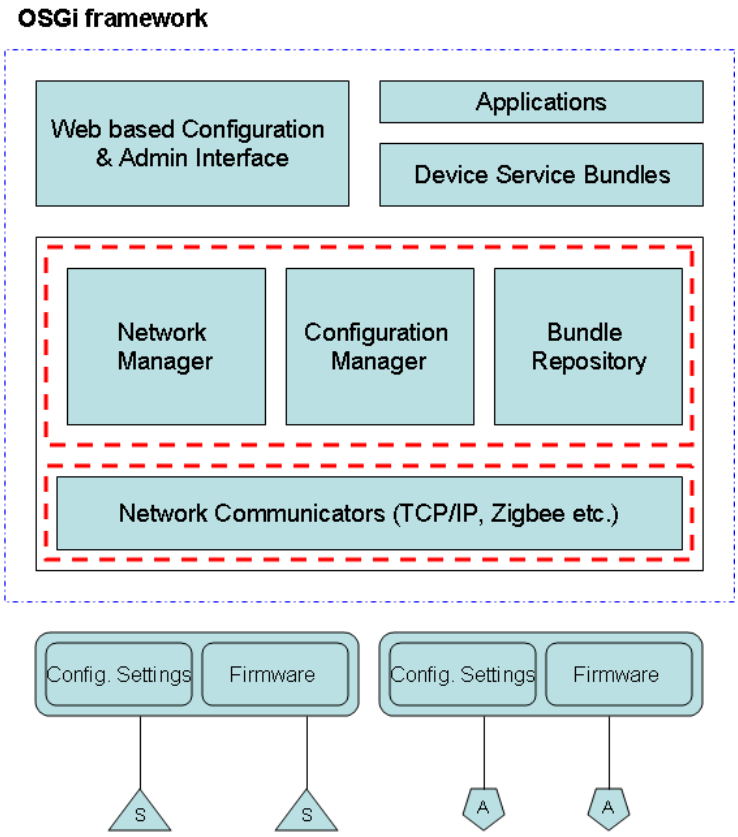
Step 3 Device automatically appears as OSGi service



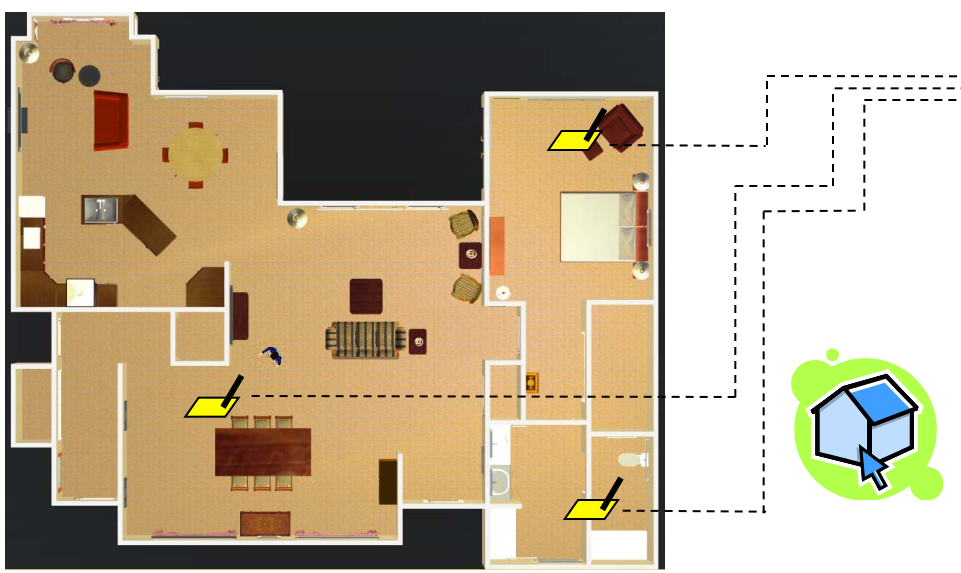

Application Layer
Atlas Reactivity Engine
Rule Composition Interface
Parser
Optimizer
Core Engine
SOA-based Programming IDE
Service Composer
Context Builder
Simulator
Services Layer
Device Service
Other Service
Query Processing
Phenomenon Detection & Tracking
Virtual Sensor
Device Service
Device Service
Atlas Middleware
Configuration Manager
Web Interface
Bundle Repository
Other Service
Network Manager
Atlas API
DDL Bundle Generator
Other Service
OSGi Framework
Internet
Node Layer
Atlas Sensor Platform
Sensor Ref
Actuator Ref
Complex Device Ref
Physical Layer
Sensors
Actuators
Complex Devices
DDL descriptor
DDL descriptor
DDL descriptor

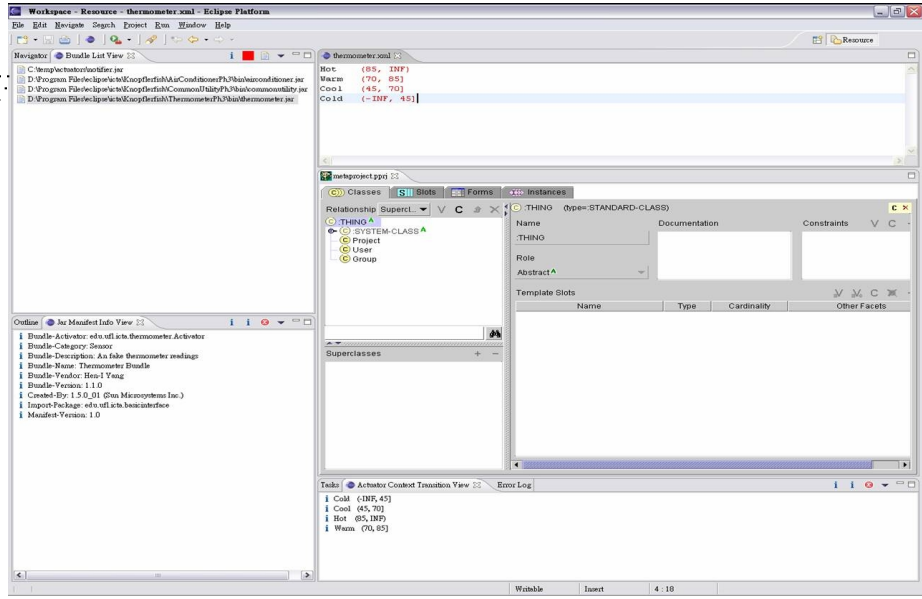
ATLAS Middleware

- Runs inside an OSGi (Open Services Gateway initiative) framework
  - Provides mechanisms for service discovery and delivery
  - Runs on a single JVM and provides a sandbox execution environment
- Core Components
  - Atlas Manager
  - Service Bundle Repository
  - Others.
- Web based Configuration & Admin Tool
  - Node net id, and middleware net address, etc.
- Eclipse Plug-in for creating smart space applications



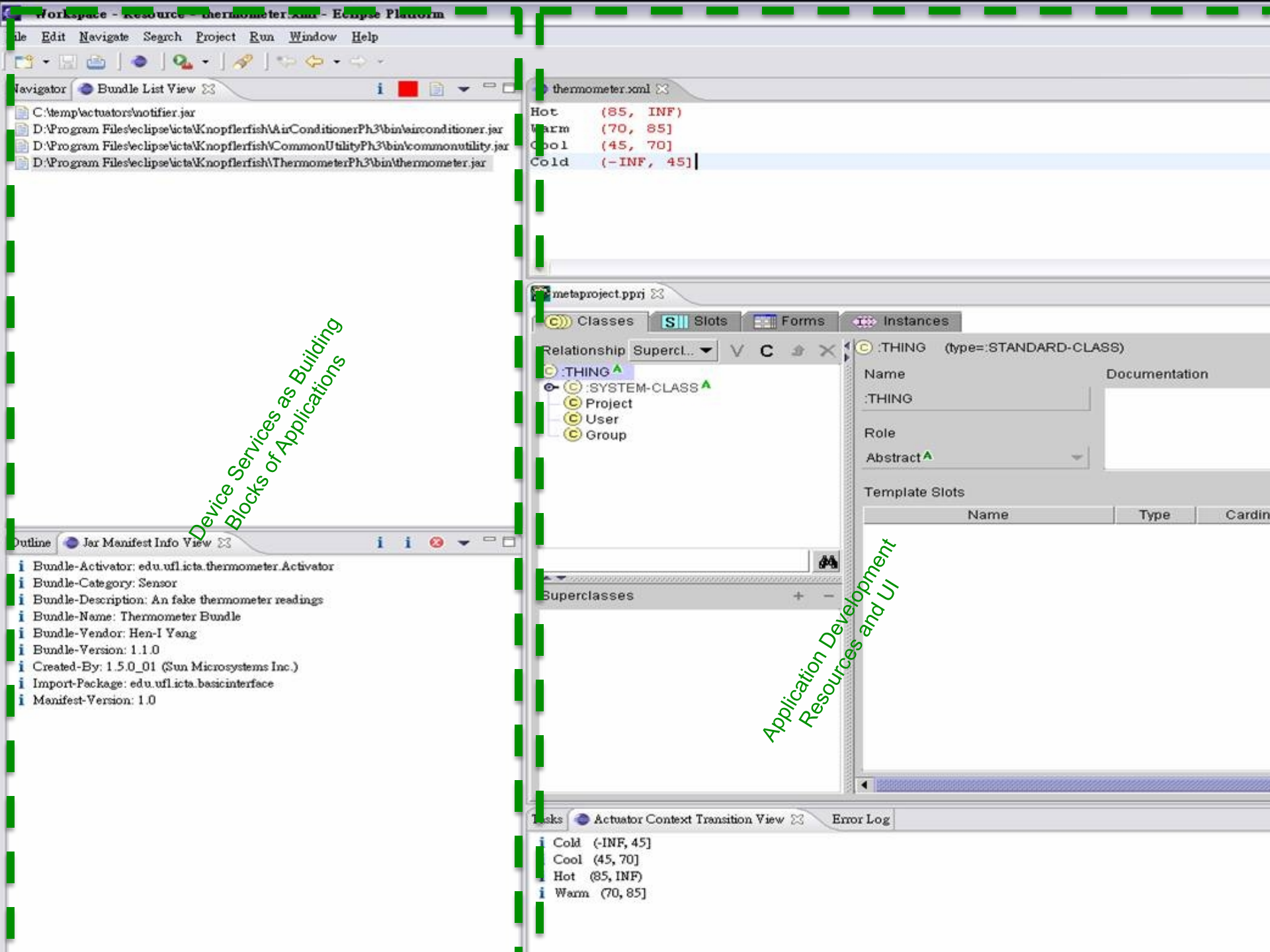
ATLAS Eclipse Plug-in Demo





3 Sensor Platforms Powered UP

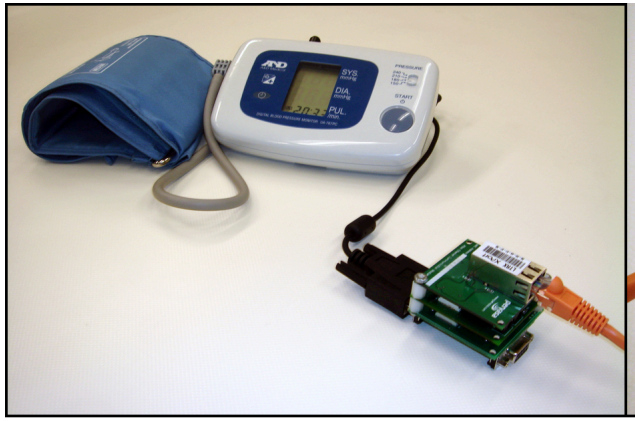
3 OSGi Service Bundles appear in the IDE

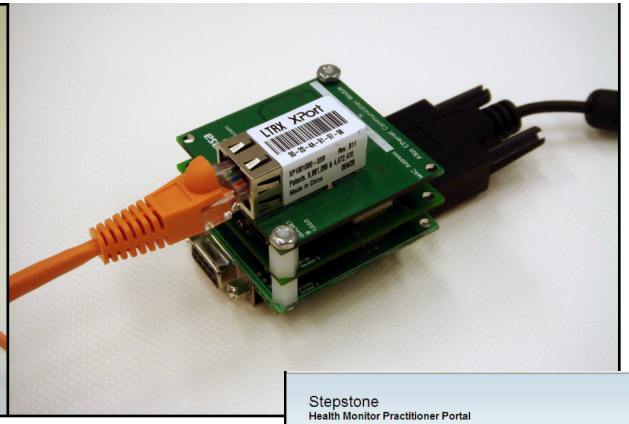

Workspace - Resource - thermometer.xml - Eclipse Platform
File Edit Navigate Search Project Run Window Help
Navigator
Bundle List View
C:\temp\actuators\notifier.jar
D:\Program Files\eclipse\icta\Knopflerfish\AirConditionerPh3\bin\airconditioner.jar
D:\Program Files\eclipse\icta\Knopflerfish\CommonUtilityPh3\bin\commonutility.jar
D:\Program Files\eclipse\icta\Knopflerfish\ThermometerPh3\bin\thermometer.jar
thermometer.xml
Hot (85, INF)
Warm (70, 85]
Cool (45, 70]
Cold (-INF, 45]
metaproject.pprj
Classes
Slots
Forms
Instances
Relationship Supercl...
:THING
:SYSTEM-CLASS
Project
User
Group
:THING (type=:STANDARD-CLASS)
Name
:THING
Documentation
Role
Abstract
Template Slots
Name
Type
Cardin
Superclasses
Device Services as Building Blocks of Applications
Application Development Resources and UI
Outline
Jar Manifest Info View
Bundle-Activator: edu.ufl.icta.thermometer.Activator
Bundle-Category: Sensor
Bundle-Description: An fake thermometer readings
Bundle-Name: Thermometer Bundle
Bundle-Vendor: Hen-I Yang
Bundle-Version: 1.1.0
Created-By: 1.5.0_01 (Sun Microsystems Inc.)
Import-Package: edu.ufl.icta.basicinterface
Manifest-Version: 1.0
Tasks
Actuator Context Transition View
Error Log
Cold (-INF, 45]
Cool (45, 70]
Hot (85, INF)
Warm (70, 85]

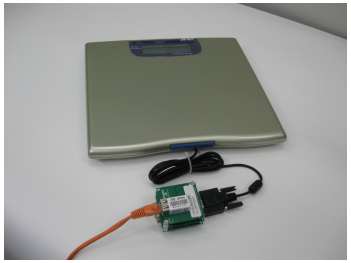
ATLAS in Use

- The Gator Tech Smart House
- The STEPSTONE Project
  - Connected Health (A Smarter World for Charlie)
- Ubiquitous Robotic Companions (joint project with ETRI, Korea).

Creating a Smarter World for Charlie using ATLAS



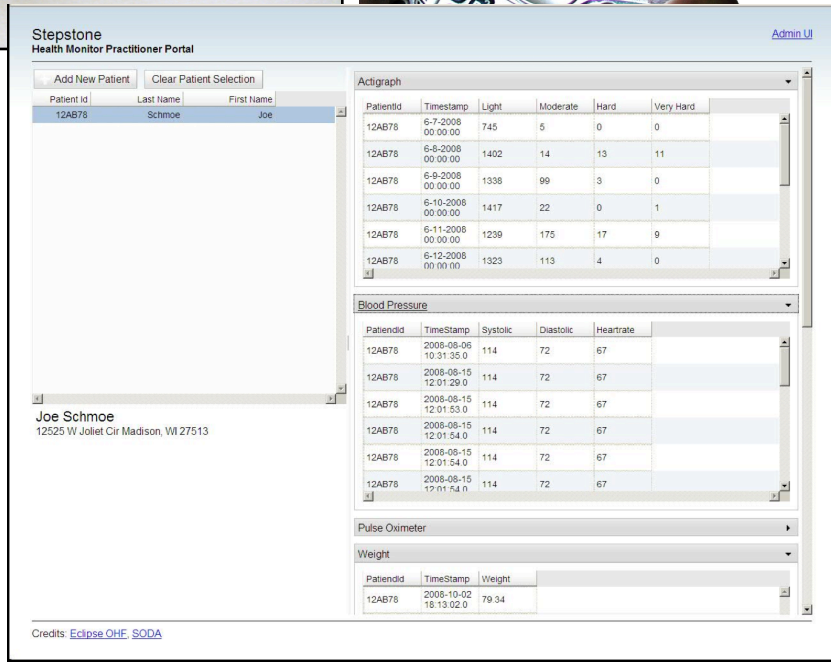




Video

UF UNIVERSITY of FLORIDA

IBM



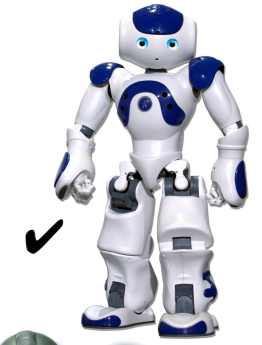
Robotic Companions as ATLAS devices in Smart Spaces

- Goals
  - Explore the use of robotic companions as a better user interface in smart spaces
  - Provide a programming environment (Smart Robot Emulator) for smart space developers to create and develop smart space interfaces easily and efficiently.

- Sponsor

IT R&D Global Leader
ETRI







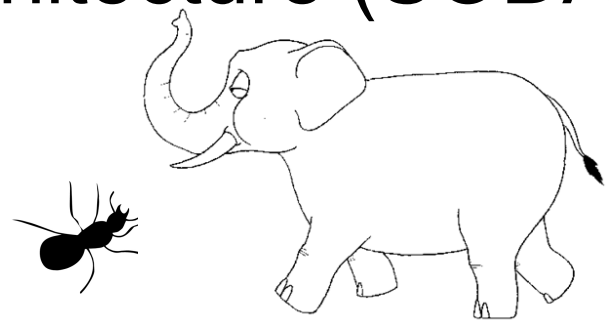
Summary: Atlas Main Contributions

- A Service Oriented Device Architecture (SODA) that enabled use of SOA as a Programming Model for Pervasive Spaces
  - Device Description Language (DDL)
  - Atlas Sensor Platform
  - Atlas Middleware

Atlas Limitations

The Ant, the Elephant, and the Monkey

- The over-promise of the Service Oriented Device Architecture (SODA)



- SODA too powerful to be safe as a programming model for pervasive spaces

Bumper Cars

Atlas Resource Page

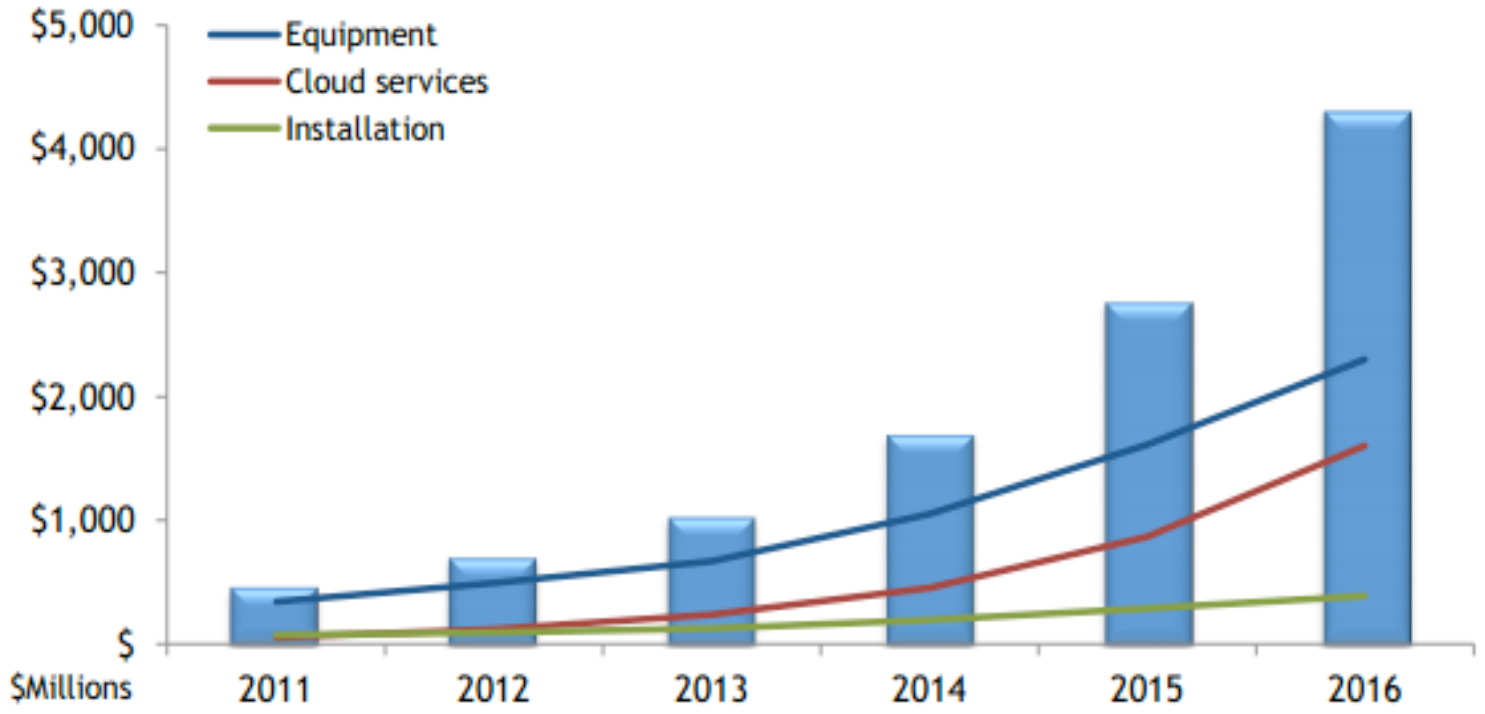
- http://www.icta.ufl.edu/atlas/

Cloud-Sensor Systems

Smart Cities – An economical pressure and an impending development lacking key Cloud-Sensor Infrastructure

U-City, Korea, A 7-years, $25B Smart City Investment

Cloud Involvement in: Energy-Smart City



Global Home Energy Management Revenues by Segment (2011-2016) – OnWorld Smart Homes, February 2012.

"Services are pressed to move to the cloud"

Introduction to Cloud-Sensor Systems

- As smart spaces and smart cities proliferate into a massive scale, sensor data and services (applications) will be pressed to move to the ***Cloud***.
  - Economies of scale and highly anticipated reductions in services costs
  - A neutral and common platform where masses of various stake holders can have access to the sensor-based services
- Dilemma! Sensors are resources that cannot be farmed and provided on demand in the cloud
- Cloud-Sensor Systems
  - Cyber-physical systems spanning the clouds and the sensors.

Challenges for Large-scale Cloud-Sensor Systems

- Cloud Scalability
  - Extensive external interactions between cloud services and the physical sensors (hundreds of millions sensors).
  - Expensive cloud “attention”, not only per sensor, but per each sensor duty cycle.

    If sensors push data once every minute, then millions of sensors will produce billions of sensor-cloud interactions, daily.
  - Requires tremendous processing power, memory resources and huge incoming/outgoing cloud traffic.
  - As a result, the cloud economies of scale per sensor will not stand.

Challenges for Large-scale Cloud-Sensor Systems

- Energy Constraint of Sensor Devices
  - Sensor devices cannot be provided dynamically and most are vulnerable to power drainage.
  - Smart city scenarios, a sensor may be queried by hundreds of applications each of which requires constant evaluation of events based on the sensor readings.
  - Deplete sensor energy rapidly. Unreliable and unavailable sensor-based services.

Cloud, Edge & Beneath (CEB)

Sensors – to – Edge; Edge – to – Cloud