MOBILE APPLICATION DEVELOPMENT
KASNEB CICT PAPER NO 15

LEARNING OUTCOMES

A candidate who passes this paper should be able to:

• Identify mobile applications, platforms and architecture
• Develop mobile applications using development tools and strategies
• Test mobile applications
• Secure mobile applications

CONTENT

15.1 Mobile devices and applications

- Definition of mobile computing
- Types of mobile devices
- Uses of mobile devices
- Overview of mobile applications
- Mobile browsers

15.2 Introduction to mobile application development

- Mobile application challenges
- Mobile application development tools
- Mobile application programming languages
- Mobile application management
- Mobile application best practices

15.3 Mobile platforms and architectures

- Internet protocols for mobile applications
- Mobile application distribution platforms and environments
- Mobile application development architectures
- Styles of mobile architecture

15.4 Mobile application development

- Mobile application development lifecycle
- Functions, arrays and objects
- Control structures and modes of execution
- Using HTML, CSS, DOM. Javasript and JQuery

15.5 iOS application development

- Window-based application and MUC
- Objective-C programming
- User Interface Design
- Introduction to graphics on the iPhone

- Core data and localisation
- Multi-threading and multi-tasking
- Web services and networking

15.6 Android application development

- Java reviews
- Androids SPK
- Resources. views and intents
- Intents and storage
- Storage and threads

15.7 Mobile application testing

- Merits and demerits of mobile application testing
- Challenges of mobile application testing
- Types of mobile application testing
- Testing tools

15.8 Mobile application security

- Reducing mobile risks
- Cloud based assessments and solutions
- Security strategies
- Security testing techniques and certification

15.9 Emerging issues and trends

TOPIC 1

15.1 Mobile devices and applications

- ***Definition of mobile computing***

Mobile computing is a generic term used to refer to a variety of devices that allow people to access data and information from where ever they are.
Also Known As: mobile device
Examples: Mobile computing can use cell phone connections to make phone calls as well as connecting to the Internet.

- ***Types of mobile devices***

We define mobile devices as having:

- the ability to connect to the Internet (or other data network)
- supports user input and interaction
- offers multiple functionalities
- is lightweight and is less than 10"

Devices that are "mobile devices":

- smartphone (and some feature phones)
- tablets
- netbooks and ultraportable laptop
- personal digital assistant (e.g. iPod Touch)
- GPS navigation device (a.k.a. car or personal navigation device)

Some participants used the "other" field to answer laptops and e-Readers. I also considered whether portable game consoles and digital audio guides (as some museums use) should be considered mobile devices.

Wikipedia's definition is pretty broad. To them a mobile is "small, hand-held computing device, typically having a display screen with touch input and/or a miniature keyboard and less than 2 pounds (0.91 kg)". Wikipedia lists calculators, digital cameras, and MP3 players as mobile device. I normally love Wikipedia but I think they are stretching the term to mean pretty much any portable electronic device.

Perhaps these are all just types of handheld computing devices. I think my definition fits the core functionality of what a device needs as a category term.

So although all the devices mentioned so far have some computing power and many have network connectivity (as even many e-Readers and digital cameras now have). But an e-Reader and digital camera are pretty much single function devices. The Kindle e-Reader does have the cool ability to of user interaction in the ability to highlight passages of eBooks and share them

online with others, but users can't create substantial content and it essentially it is a single function device (hence the name even).
Laptops may be portable but they aren't portable enough to allow ubiquitous access - a trait that I think is central to the concept of mobile device (opposed to just portable device).

- ***Uses of mobile devices***

Handheld devices have become ruggedized for use in mobile field management. Uses include digitizing notes, sending and receiving invoices, asset management, recording signatures, managing parts, and scanning barcodes.

Recent developments in mobile collaboration systems employ handheld devices that combine video, audio and on-screen drawing capabilities to enable multi-party conferencing in real-time, independent of location.

Handheld computers are available in a variety of form factors, including smartphones on the low end, handheld PDAs, Ultra-Mobile PCs and Tablet PCs (Palm OS, WebOS).

Users can watch television through Internet on mobile devices. Mobile television receivers have existed since the 1960s, and in the 21st century mobile phone providers began making television available on cellular phones.

Nowadays, mobile devices can create, sync, and share everything we want despite of distance or specifications of mobile devices. In the medical field, mobile devices are quickly becoming essential tools for accessing clinical information such as drugs, treatment, and even medical calculation.

Due to the popularity of Candy Crush and other mobile device games, online casinos are also offering casino games on mobile devices. The casino games are available on iOS, Android, Windows Phone and Windows. Available games are roulette, blackjack and several different types of slots. Most casinos have a play for free option.

In the military field, mobile devices have created new opportunities for the Army to deliver training and educational materials to soldiers around the world.

- ***Overview of mobile applications***

Mobile application development is a term used to denote the act or process by which application software is developed for handheld devices, such as personal digital assistants, enterprise digital assistants or mobile phones. These applications can be pre-installed on phones during manufacturing platforms, or delivered as web applications using server-side or client-side processing (e.g. JavaScript) to provide an "application-like" experience within a Web browser.

Application software developers also have to consider a lengthy array of screen sizes, hardware specifications and configurations because of intense competition in mobile software and changes within each of the platforms. Mobile app development has been steadily growing, both in terms of revenues and jobs created. A 2013 analyst report estimates there are 529,000 direct App Economy jobs within the EU 28 members, 60% of which are mobile app developers.

As part of the development process, Mobile User Interface (UI) Design is also an essential in the creation of mobile apps. Mobile UI considers constraints & contexts, screen, input and mobility as outlines for design. The user is often the focus of interaction with their device, and the interface entails components of both hardware and software. User input allows for the users to manipulate a system, and device's output allows the system to indicate the effects of the users' manipulation. Mobile UI design constraints include limited attention and form factors, such as a mobile device's screen size for a user's hand(s). Mobile UI contexts signal cues from user activity, such as location and scheduling that can be shown from user interactions within a mobile application.

Overall, mobile UI design's goal is primarily for an understandable, user-friendly interface. The UI of mobile apps should: consider users' limited attention, minimize keystrokes, and be task-oriented with a minimum set of functions. This functionality is supported by Mobile enterprise application platforms or Integrated development environments (IDEs).

Mobile UIs, or front-ends, rely on mobile back-ends to support access to enterprise systems. The mobile back-end facilitates data routing, security, authentication, authorization, working off-line, and service orchestration. This functionality is supported by a mix of middleware components including mobile app servers, Mobile Backend as a service (MBaaS), and SOA infrastructure.

Contents

Platform

The platform organizations need to develop, deploy and manage mobile apps is made from many components, and tools allow a developer to write, test and deploy applications into the target platform environment.

Front-end development tools

Front-end development tools are focused on the user interface and user experience (UI/UX) and provide the following capabilities:

- UI design tools
- SDKs to access device features
- Cross-platform accommodations/support

Back-end servers

Back-end tools pick up where the front-end tools leave off, and provide a set of reusable services that are centrally managed and controlled and provide the following capabilities:

- Integration with back-end systems
- User authentication/authorization
- Data services
- Reusable business logic

Security add-on layers

With BYOD becoming the norm within more enterprises, IT departments often need stop-gap, tactical solutions that layer on top of existing apps, phones, and platform component. Features include

- App wrapping for security
- Data encryption
- Client actions
- Reporting and statistics

System software

There are many system-level components that are required to have a functioning platform for developing mobile apps.

Criteria for selecting a development platform usually contain the target mobile platforms, existing infrastructure and development skills. When targeting more than one platform with cross-platform development it is also important to consider the impact of the tool on the user experience. Performance is another important criterion, as research on mobile applications indicates a strong correlation between application performance and user satisfaction. In

addition to performance and other criteria, the availability of the technology and the project's requirement may drive the development between native and cross-platform environments. To aid the choice between native and cross-platform environments, some guidelines and benchmarks have been published. Typically, cross-platform environments are reusable across multiple platforms, leveraging a native container while using HTML, CSS, and JavaScript for the user interface. In contrast, native environments are targeted at one platform for each of those environments. For example, Apple iOS applications are developed using Xcode with Objective C and/or Swift, Android development is done in the Eclipse IDE with the ADT (Android Developer Tools) plugins, and Windows and BlackBerry also have their own development environment.

Mobile application testing

Mobile applications are first tested within the development environment using emulators and later subjected to field testing. Emulators provide an inexpensive way to test applications on mobile phones to which developers may not have physical access. The following are examples of tools used for testing application across the most popular mobile operating systems.

- **Google Android Emulator** - Google Android Emulator is an Android emulator that is patched to run on a Windows PC as a standalone app, without having to download and install the complete and complex Android SDK. It can be installed and Android compatible apps can be tested on it.
- **The official Android SDK Emulator** - The official Android SDK Emulator includes a mobile device emulator which mimics all of the hardware and software features of a typical mobile device (without the calls).
- **MobiOne** - MobiOne Developer is a mobile Web IDE for Windows that helps developers to code, test, debug, package and deploy mobile Web applications to devices such as iPhone, BlackBerry, Android, and the Palm Pre.
- **TestiPhone** - TestiPhone is a web browser-based simulator for quickly testing iPhone web applications. This tool has been tested and works using Internet Explorer 7, Firefox 2 and Safari 3.
- **iPhoney** - iPhoney gives a pixel-accurate web browsing environment and it is powered by Safari. It can be used while developing web sites for the iPhone. It is not an iPhone simulator but instead is designed for web developers who want to create 320 by 480 (or 480 by 320) websites for use with iPhone. iPhoney will only run on Mac OS X 10.4.7 or later.
- **BlackBerry Simulator** - There are a variety of official BlackBerry simulators available to emulate the functionality of actual BlackBerry products and test how the BlackBerry device software, screen, keyboard and trackwheel will work with application.
- **Windows UI Automation** - To test applications that use the Microsoft UI Automation technology, it requires Windows Automation API 3.0. It is pre-installed on Windows 7, Windows Server 2008 R2 and later versions of Windows. On other operating systems, you can install using Windows Update or download it from the Microsoft Web site.

Tools include

- eggPlant: A GUI-based automated test tool for mobile application across all operating systems and devices.
- Ranorex: Test automation tools for mobile, web and desktop apps.
- Testdroid: Real mobile devices and test automation tools for testing mobile and web apps.

Application stores

Several initiatives exist both from mobile vendor and mobile operators around the world. Application developers can propose and publish their applications on the stores, being rewarded by a revenue sharing of the selling price. An example is Apple's App Store, where only approved applications may be distributed and run on iOS devices (otherwise known as a walled garden). There are approximately 700,000 iOS Applications. Google's Android Market (now known as the "Play Store") has a large number of apps running on devices with Android OS. HP / Palm, Inc have also created the Palm App Catalog where HP / Palm, Inc webOS device users can download applications directly from the device or send a link to the application via a web distribution method. Mobile operators such as Telefonica Group and Telecom Italia have launched cross-platform application stores for their subscribers. Additionally, mobile phone manufacturers such as Nokia has launched Ovi app store for Nokia smartphones. Some independent companies, namely Amazon Appstore, Aptoide and GetJar, have created their own third-party platforms to reach more users in different locations. The Windows Phone Marketplace had more than 100,000+ apps available as of 7-11-2012.

Patents

There are many patents applications pending for new mobile phone apps. Most of these are in the technological fields of Business methods, Database management, Data transfer and Operator interface.

On May 31, 2011, Lodsys asserted two of its four patents: U.S. Patent No. 7,620,565 ("the '565 patent") on a "customer-based design module" and U.S. Patent No. 7,222,078 ("the '078 patent") on "Methods and Systems for Gathering Information from Units of a Commodity Across a Network." against the following application developers:

- Combay
- Iconfactory
- Illusion Labs
- Shovelmate
- Quickoffice
- Richard Shinderman of Brooklyn, New York
- Wulven Game Studios of Hanoi, Vietnam

- ***Mobile browsers***

A **mobile browser** is a web browser designed for use on a mobile device such as a mobile phone or PDA. Mobile browsers are optimized so as to display Web content most effectively for small screens on portable devices. Mobile browser software must be small and efficient to accommodate the low memory capacity and low-bandwidth of wireless handheld devices. Typically they were stripped-down web browsers, but some more modern mobile browsers can handle more recent technologies like CSS 2.1, JavaScript, and Ajax.

Websites designed for access from these browsers are referred to as *wireless portals* or collectively as the Mobile Web. They may automatically create "mobile" versions of each page.

DEFINATION 2:
A mobile browser is one that is optimized for the small display screen and limited resources of a handheld computing device such as a smart phone. A mobile browser interface is simplified to display content in the smallest viable space. The browser software is as lightweight as possible to address memory and bandwidth constraints.
Mobile browsers connect to the Internet through a cellular network or a wireless LAN (local area network). Some mobile browsers can display regular HTML sites, while others can only display websites that have been specially formatted for mobile browsers. Content optimized for mobile browsers is typically text-based or low-graphic and may be written in languages that were designed for mobile computing such as WML (wireless markup language) or CHTML (compact HTML). Most current mobile browsers today can display regular HTML.
The first mobile browser was Apple's NetHopper, which was released in 1996 for PDAs.
Here is a list of popular mobile browsers:
Bolt, Firefox for mobile, Internet Explorer for mobile, JB5,Myriad, NetFront, Novarr Vision, Obigo, Opera, Opera Mini, Ploaris, Skyfire, uZard

TOPIC 2

15.2 Introduction to mobile application development

Mobile application development is a term used to denote the act or process by which application software is developed for handheld devices, such as personal digital assistants, enterprise digital assistants or mobile phones. These applications can be pre-installed on phones during manufacturing platforms, or delivered as web applications using server-side or client-side processing (e.g. JavaScript) to provide an "application-like" experience within a Web browser. Application software developers also have to consider a lengthy array of screen sizes, hardware specifications and configurations because of intense competition in mobile software and changes within each of the platforms. Mobile app development has been steadily growing, both in terms of revenues and jobs created. A 2013 analyst report estimates there are 529,000 direct App Economy jobs within the EU 28 members, 60% of which are mobile app developers.

As part of the development process, Mobile User Interface (UI) Design is also an essential in the creation of mobile apps. Mobile UI considers constraints & contexts, screen, input and mobility as outlines for design. The user is often the focus of interaction with their device, and the interface entails components of both hardware and software. User input allows for the users to manipulate a system, and device's output allows the system to indicate the effects of the users' manipulation.

Mobile UI design constraints include limited attention and form factors, such as a mobile device's screen size for a user's hand(s). Mobile UI contexts signal cues from user activity, such as location and scheduling that can be shown from user interactions within a mobile application.

Overall, mobile UI design's goal is primarily for an understandable, user-friendly interface. The UI of mobile apps should: consider users' limited attention, minimize keystrokes, and be task-oriented with a minimum set of functions. This functionality is supported by Mobile enterprise application platforms or Integrated development environments (IDEs).

Mobile UIs, or front-ends, rely on mobile back-ends to support access to enterprise systems. The mobile back-end facilitates data routing, security, authentication, authorization, working off-line, and service orchestration. This functionality is supported by a mix of middleware components including mobile app servers, Mobile Backend as a service (MBaaS), and SOA infrastructure.

- ***Mobile application challenges***

Over the next few years, improving the convenience of mobile services will depend on improving the use of context in delivering mobile experiences. Your business will need to predict what your customers want when they launch a mobile application or website. Delta

Airlines, for example, knows how close a passenger is to departure time and delivers relevant content, such as a frequent flier's real-time status on the upgrade list for her next flight. Today, that kind of context is uncommon. Tomorrow, it will be table stakes.
Application developers writing mobile apps will have to start thinking about "mobile context" which we define as everything your customer has told you and all you can understand about what the customer is currently experiencing. Context is just one of the big, new challenges that application developers will face. Here are several more of the most important challenges we see.

1. Context.
Your customer's mobile context consists of:
Preferences: The history and personal decisions the customer has shared with you or with social networks.
Situation: The current location, of course, but other relevant factors could include the altitude, environmental conditions and even speed the customer is experiencing.
Attitude: The feelings or emotions implied by the customer's actions and logistics.
Delivering a good contextual experience will require aggregating information from many sources. It could be from the devices customers are carrying, the local context of devices and sensors around them (e.g. a geofence that knows which airport gate they're at), an extended network of things they care about (e.g. the maintenance status of the incoming airplane they are about to take for their next flight, and the probability it will leave on time) and the historical context of their preferences. Gathering this data is a major challenge because it will be stored on multiple systems of record to which your app will need to connect.

2. Device Proliferation.
Another challenge facing mobile developers is device proliferation. It might seem like today's mobile app development process is pretty well defined: Build your app, make sure it looks pretty on a 4-inch smartphone and a 10-inch tablet, and then submit it to an app store. It's not quite that easy now, and it'll be much tougher in the near future. A wide range of new device sizes and changes to the nature of the apps themselves will increase the need for flexibility, especially on the client. We're already seeing 5-inch phablets, 7-inch tablets, and Windows 8 devices of 20 inches or more. Collectively, these new devices will significantly expand the potential for collecting contextual data about your customers. Here are some ideas of what changes you'll face:

3. Voice, Prioritized Over Touch.
Mobile developers are clamoring for API access to Apple's Siri and Google Now. There are a lot of scenarios where you would want to build voice input into your app today. For a running or fitness app, a phone is likely to be strapped to a person's sweaty arm, and looking at your screen while running can be a fast track into a lamp post. The same is true while driving. If you're hustling through an airport with luggage to catch a flight, voice beats touch. Modern applications will let people use their devices while keeping their eyes and hands off it.

4. Heads-Up Interfaces.

Expect to see heads-up displays such as Google Glass go main stream in the next five years as Moore's law pushes processors to the point where such gadgets can be made powerful, lightweight and perhaps even stylish. Augmented-reality apps that don't work well on a phone or tablet could be transformative when ported to a device like Google Glass. A compelling example would be an app that provides real-time information about the people you are talking to but whose names you've forgotten.
But heads-up displays will create a whole new slate of problems for developers. We'll have to adapt to peripheral cues such as reminders and alerts that don't block the user's vision. We'll also need to integrate tactile and aural feedback such as voice commands and vibrating sensors that alert users they need to take action.

5. Bigger -- And Smaller -- Touch Devices, And Adaptive UIs.
Today, most app developers prioritize a few popular devices, such as the iPhone, the Samsung Galaxy S III and the iPad. But cherry picking the most popular devices will become more of a challenge as device types and platforms proliferate. Google and Apple already support tablets of different sizes and, with Windows 8 now shipping, developers can expect to find a whole range of larger touch-sensitive devices, such as Hewlett-Packard's Envy series. But device surfaces will grow beyond specialized devices as the cost of multi-touch monitors falls -- to the point where touchwall computing becomes broadly available. Developers will need to scale their user interfaces, because an 84-inch experience is very different from a 4-inch experience.

6. Mobile Apps Become Pluggable Mobile Services.
Platform vendors such as Apple and Google are offering more platform-specific services that developers can leverage. Apple Passbook and Google Wallet are already established, but other examples such as Microsoft's Windows Phone 8 hubs and the Blackberry 10 "Peek" and "Flow" user interface further erode the distinction between a mobile platform and the apps that run on it. Over the next few years it will become more difficult to tell where the mobile platform services end and the third-party app begins. Expect mobile platform providers to relentlessly push toward device-integrated, client-side services instead of standalone mobile apps, because platform providers force developers to tailor apps to their unique platform and APIs.

7. Wearables, Connectables and Local Networks.
Simple, first-generation wearables such as the Nike+ FuelBand and Fitbit will give way to something much more practical: internal biomedical instruments like pacemakers or insulin pumps. Imagine shoes that turn steps into power that can recharge devices, and golf clubs that provide swing telemetry that can help a player improve their game. Connectable home alarm systems, automobiles and scales all will provide data that can be viewed on a mobile phone or tablet, turning it into a remote control for an Internet of things. Client-side developers will need to release updates for these connected devices faster than ever to dynamically add support for new devices in the extended local network.

8. Hybrid Application Model.
With each release, popular mobile operating systems get better at supporting HTML5 and its attendant APIs. That capability will let companies reuse more code across multiple devices,

which will be important in keeping app development costs down amid the proliferation of connected devices and form factors. Supporting native development on three or more mobile platforms is prohibitively expensive, and it's difficult to support feature development in parallel across multiple code bases.
As a result, more organizations will turn to Web technologies with a centralized code base for core components of their connected applications. That centralized code is likely to be a combination of HTML, JavaScript and CSS, because the vast majority of devices will support it. Much of this HTML5 will be delivered as hybrid apps, which mix HTML code into a native container and use native code for navigation and platform specific integration. These hybrid apps will get more sophisticated and capable as a result. We're already seeing compelling enterprise applications built with the hybrid approach, and even consumer apps built as hybrids are showing up as editor's choices and with 4+ star ratings.

9. Cloud-Powered Development.
The construction of modern applications will move onto the public cloud and public devices, because the elasticity of services such as Amazon, Microsoft Azure and the Google Cloud Platform mesh nicely with the unpredictable demand that mobile apps exact on server-side infrastructure. With the move to a public cloud, the traditional organizational model that separates development from IT operations will break down. Why? Mobile development requires a rapid feedback cycle, and it's hard to execute if developers have to wait for IT operations teams to respond to their change requests. Developer self-provisioning will re-balance the relationship in favor of developers, away from traditional IT organizations -- because control over hardware and infrastructure resources will no longer be absolute.
But with greater developer power comes greater responsibility for security and performance. Expect more developers to be on call for application support in the new model, using triage to handle defects and investigate degradation to production services. Those tasks have traditional been the domain of systems administrators. Expect IT operations personnel to become integrated into development teams and to start their work at the inception of an idea.

- ***Mobile application development tools***

10 low-code tools for building mobile apps fast

While debate rages on among various mobile development camps, businesses still have to create and maintain mobile apps for their employees, business partners, and customers. The pure HTML5/JavaScript/CSS3 mobile Web faction, the native-code purists, the hybrid mobile app fans -- they all offer compelling arguments and approaches, but the one conclusion everyone seems to reach, eventually, is that there is no single panacea. Each approach and tool set has advantages and drawbacks.
The difficulty and cost of mobile app development has not escaped the notice of innovative companies. We present here 10 low-code or no-code builders for mobile applications. Some target more than one mobile platform, some target Web applications as well, but all are aimed at getting your organization’s mobile project up and running quickly.

1 Alpha Anywhere

A low-code, rapid, wizard-driven, end-to-end builder with a Windows-based IDE, Alpha Anywhere supports many databases and targets Web, mobile (iOS, Android, and Windows Phone), and desktop applications. HTML apps can be built using a component-based designer and responsively adapt to screen sizes from 4 inches to 4 feet. Alpha Anywhere integrates with PhoneGap and Adobe PhoneGap Build, allowing the easy creation of hybrid mobile apps without requiring the developer to install multiple native development environments or purchase a Mac. The company is currently testing a unique solution for occasionally connected mobile apps that rely on remote databases.

2 App Press

App Press is a Web-based no-code app creator that targets iPhone, iPad, and Android applications. Geared for designers, App Press uses a Photoshop-like user interface for assembling screens from visual assets using layers. On the back end, App Press is an Amazon cloud-based service and platform. The company claims that designers can produce their first app in one day, that with experience designers can create five apps a day, and that experienced designers can train new designers on the platform.

3 AppArchitect

AppArchitect is a Web-based, no-code; drag-and-drop builder and platform for native iPhone and iPad apps, which can be previewed in the AppArchitect Preview App, downloadable from the iTunes App Store, and finished binaries can be downloaded to submit to the App Store. It assembles plug-in building blocks that are written in Objective-C, and an AppArchitect SDK will be available to extend the product's capabilities. The company plans to expand the product to generate Android and mobile Web apps in the future, and it plans to charge $40 to $100 per month once the product is released.

4 Form.com

Form.com is a Web-based enterprise platform for Web and mobile form solutions with a drag-and-drop forms builder and flexible back-end technology. The builder can create new forms or replicate existing paper forms, set up process-specific workflow and API integration, embed logical transitions, allow the capture of images within the forms, capture digital signatures, and enable form field autofill. Finished mobile forms can collect information when disconnected and transfer it when connection has been restored.

5 iBuildApp

iBuildApp is a Web builder that offers customizable templates for iPhone, iPad, and Android apps and promises that you can create an app in five minutes. Your app can be free if you accept iBuildApp branding and very tight limits to the number of users and site visits, unlimited-user white-labeled tablet apps cost $299 a month, and there are several plans in between the extremes. For common app types, template-based systems like iBuildApp can sometimes produce usable results, as long as the selection of widgets includes the functionality you need.

6 QuickBase

QuickBase is an online builder and platform for Web and mobile Web database applications. It offers more than 300 customizable application templates, including the Complete Project Manager shown in the slide. Users can build applications "from scratch" starting with a data design and all QuickBase websites can also be viewed as mobile websites. While Mobile QuickBase is not currently available in app form, the mobile website is eminently usable.

7 Salesforce1

Salesforce1 gives you the ability to accelerate the development and deployment of HTML5, iOS, and Android mobile apps, as well as Web apps. In the simplest model, you use a mobile website or downloadable generic Salesforce viewer app to work with your Force.com Web application. One step up from that is to create a jQuery Mobile (shown in the slide), Angular.js, Backbone.js, or Knockout HTML5 mobile app using a Salesforce Mobile Pack. At the most complicated level, you can create native or hybrid apps for iOS and Android using the Salesforce Mobile SDK for your mobile platform combined with the Native SDK tools. These apps all communicate with the back end through a Connected App in Salesforce.

8 ViziApps

ViziApps combines an online visual designer and customizable sample apps with code generation for mobile Web, as well as iOS and Android native apps. The ViziApps designer has form fields and charts, 60 backgrounds, and 4,000 stock images. It supports maps, video, audio, navigation bars, and navigation panels, and it has lots of customizations and JavaScript extensions. Template apps show how fields, actions, and data interfaces are used.

9 Mobile Chrome Development Kit

The Mobile Chrome Development Kit, recently released as a Developer Preview tool chain based on Apache Cordova, takes a hybrid app strategy. A single project targets iOS, Android, and Chrome apps. The user interface is standard HTML and CSS, which is integrated into Android (shown) and iOS native toolkits. While this is definitely *not* a no-code tool, you can do a lot using any visual HTML page designer. Once you need to add mobile code, you have Chrome APIs and Cordova APIs at your disposal, from JavaScript to enhance the app without having to drop down to the platform-dependent native code level.

10 Appcelerator

Appcelerator combines an IDE, SDK, multiple frameworks, and back-end cloud services into an enterprise-level system for mobile development. The Titanium SDK lets you develop native, hybrid, and mobile Web applications from a single codebase.

11 Titanium Studio

Is an extensible, Eclipse-based IDE for building Titanium and Web apps, and Appcelerator Cloud Services provide an array of automatically scaled network features and data objects for your app. The Alloy framework is an Appcelerator framework designed to rapidly develop Titanium applications, based on the MVC architecture and containing built-in support for Backbone.js

and Underscore.js. While Appcelerator is *not* a no-code solution, it provides JavaScript-based tooling for iOS, Android, Tizen, BlackBerry, and mobile Web applications in one place.

- ***Mobile application programming languages***

One of the very first steps in the app development process is choosing which programming language to use. It seems like a simple decision, but different operating systems favor different programming languages. If you want to immerse yourself in the app development world, below are the top 5 programming languages that you should learn (or review if you are already senior developer).

JavaScript:
JavaScript is probably the most common and most recognizable of the programming languages needed for app development. It is used extensively in web browsing, and it has made the transition to the mobile world. JavaScript is beneficial because it can be used across a variety of platforms without much difficultly. For junior developers, Javascript is a relatively simple and extremely useful programming language to master. It is important to keep in mind that there is no official universal standard for Javascript, so it may be rendered differently across different platforms.

Java:
Not to be confused with Javascript, Java is object-oriented programming language that is platform independent (meaning it can be used across different operating systems), but it is used extensively with Google's Android mobile operating system. Object-oriented programming languages are organized around objects and data rather than logic and actions. Java works by categorizing objects and data together based on similar function as well as similar properties. Because it shares a similar structure with basic C-based languages, Java is a great transition language for intermediate developers because the syntax is much simpler than languages like C++ and there are extensive libraries for beginners.

C#:
C# (pronounced C-sharp) is the default (and recommended) programming language for Windows-based apps. With Windows Phone poised to make a comeback with Windows 10 Mobile, and the Windows App Store still desperately in need of well-made apps, learning C# could give you a leg up in the Windows marketplace. C# is an object-oriented programming language like Java, and it is based on the classical C-type languages. If you have a background in basic programming languages, C# shouldn't be hard to pick up.
C# plays the role in the Microsoft universe as the Objective-C plays in the Apple cosmos: It's an expansion of C that directly addresses many of the unique features of the environment. The Windows Mobile platform hasn't been the market-changer that many had predicted (and hoped), but there's no denying the gravitational pull of Windows across multiple platforms. If your fleet of mobile devices includes Windows then your suite of development languages should include C#.

Swift:

Created by Apple, Swift was introduced at 2014 WWDC Apple showcase. Swift is a multi-paradigm, compiled programming language designed to work with Apple's iOS and OS X systems. Swift is meant to be easier to learn and less bug-prone than Objective-C, but it works with Apple's Cocoa/Cocoa Touch frameworks, as well as existing Objective-C code, without issue. Swift was developed with the idea of creating fast, high-performing apps simply and easily.

Apple's latest APIs are Cocoa and Cocoa Touch. The language to write code for them is Swift. According to Apple, Swift is written to work along with Objective-C, though it's obvious that the company intends for many developers to turn to Swift for complete programming. Among other things, Swift has been designed to eliminate the possibility for many of the security vulnerabilities possible with Objective-C. If you're now beginning the process of writing iOS apps, then Swift should be your starting point. If you've been developing apps for iOS, then it's time to start training your developers on Swift.

As an additional justification for Swift, at WWDC 2015 Apple announced that Swift will be going open source this fall. That's bound to increase the number of people willing to work with Swift and increase the number of projects for which Swift becomes the primary development language

PHP:

PHP is a server-side programming language which shares similar syntax with other C-based programming languages, making it easy to pick up for C-based developers. PHP supports a large range of database types, making it ideal for any application that needs access to a database. PHP is also extremely flexible, allowing it to support object-oriented programming languages but it can also function well without them. PHP is a great choice for creating the interfaces for mobile applications, and PHP is very useful for simplifying the codes and functions of other languages. Compared to other languages, PHP applications do tend a run a bit slower than others. But, as PHP is open-source, improvements are being made constantly.

HTML5

If you want to build a Web-fronted app for mobile devices, the one near-certainty is HTML5. The eventual standard will make various data types simple to insert, rationalize input parameters, level the browser playing field, account for different screen sizes, and probably freshen your breath and give you lush, manageable hair.

The problem is that HTML5 is still a proposed standard that is currently supported in a lot of different ways by a lot of different browsers. It's certainly possible to write HTML5 Web pages now, and many people are doing just that. They just have to know that there might be slight tweaks in the language in months to come and more substantial changes in the way browsers handle HTML5.

From a cost and efficiency standpoint HTML5 has the advantage of building on the current version of HTML so the learning curve is much shallower than that for a completely new language. If you can cope with a bit of uncertainty and want to walk the browser-based path, HTML5 is an obvious choice for a primary language.

Objective-C
While most of the world was developing software using C++, Apple went with Objective C as its primary programming language. Like C++, Objective C is a C-language superset. It does many of the same things for C that C++ does, though it has a number of functions that specifically deal with graphics, I/O, and display functions. Objective-C is part of the Apple development framework and is fully integrated into all iOS and MacOS frameworks. It is in the process, though, of being replaced in the Apple ecosystem -- by Swift.

Which to Choose?
So which language is the "best" for mobile development? As with so many other topics in the software development world, the best answer is "it depends." If you want to do native development on iOS, your hand is forced. If you want to build an app with a browser front-end, have rich media as part of the experience, and would like to have your app relatively future-proofed, then there's only one real choice. For everything else, you'll have to look at the experience in your staff, the needs of your users, and the budget for the project.
People tend to be invested in languages and systems. Which would you choose? Where have you made your mobile development investment? I'd love to hear what you have to say.

- ***Mobile application management***

Mobile application management (**MAM**) describes software and services responsible for provisioning and controlling access to internally developed and commercially available mobile apps used in business settings on both company-provided and “bring your own” smartphones and tablet computers (BYOD).
Mobile application management differs from **mobile device management** (MDM). As the names suggest; MAM focuses on application management, it provides a lower degree of control over the device, but a higher level of control over applications. MDM solutions manage the down to device firmware and configuration settings and can include management of all applications and application data.

App wrapping
App wrapping was initially a favored method of applying policy to applications as part of mobile application management solutions.
App wrapping sets up a dynamic library and adds to an existing binary that controls certain aspects of an application. For instance, at startup, you can change an app so that it requires authentication using a local passkey. Or you could intercept a communication so that it would be forced to use your company's virtual private network (VPN) or prevent that communication from reaching a particular application that holds sensitive data.
Increasingly, the likes of Apple and Samsung are overcoming the issue of app wrapping. Aside from the fact that app wrapping is a legal grey zone, and may not meet its actual aims, it is not possible to adapt the entire operating system to deal with numerous wrapped apps. In general, wrapped apps available in the app stores have also not proven to be successful due to their inability to perform without MDM.

System features

An end-to-end MAM solution provides the ability to: control the provisioning, updating and removal of mobile applications via an enterprise app store, monitor application performance and usage, and remotely wipe data from managed applications. Core features of mobile application management systems include:

• App delivery (Enterprise App Store) • App updating • App performance monitoring • User authentication • Crash log reporting • User & group access control	• App version management • App configuration management • Push services • Reporting and tracking • Usage analytics • Event management • App wrapping

- ***Mobile application best practices***

With the advent of mobile devices, a new industry came into existence. Mobile devices are now so popular that many users no longer buy desktop or laptop computers. Advertisers, seeing the value of this new medium are taking for advantage of it, offering products, games, apps and more. In this article you'll learn about 10 design practices for building mobile apps. These practices will help you get the results you seek and also satisfy your customers.

1. **Before You Begin, Consider Your Audience:** Before you take any time to build an app, consider your audience. What do you hope to achieve? How do you envision your audience using your app? These are important questions to consider up-front.

2. **Check the App Stores:** Many times people come up with a great idea for an app and start to brainstorm how to build it. There's only one problem. Despite how unique you might think your idea is, there's an excellent chance that someone might have already built it, or something similar to it. If that's the case, you would be wasting a ton of time (and money). If an app already exists, you can use it as a template to create your own product, or you might consider partnering with the creator(s) of that app and using it as part of your strategy.

3. **Involve Potential Users in the Design Process:** One danger of any design process is working only with your team and not involving the end users at all. Then, when the design is done and is released to the public, some or many aspects of your design might not translate well to the real world. To avoid this problem, involve potential end users in the design process and use their feedback to make changes as necessary.

4. **Create a Storyboard:** The storyboard is one of the most important aspects of the design process. This is where you lay out the complete functionality of your app on paper. If there are problems, you can resolve them at this stage. The storyboard allows you to plan out all aspects of the design, including future components, such as plug-ins.

5. **Make the App Easy to Understand:** The app should be easy to understand with descriptions to accompany graphics (if necessary) and additional instructions. One design flaw is relying too much on images to tell the tale. That's a major error because users might not be able to figure out the purpose of your app if you use a lot of graphics. Clear instructions are necessary.

6. **Avoid Overuse of Graphics and Animations:** Both graphics and animations can add a nice "Wow" factor to your app but there's a major downside – slow loading times which translate into a poor user experience. Whenever possible, either avoid the use of bitmaps or animations or limit their use to only essential features. And if you do use graphics, use vector graphics whenever possible. The files sizes from these are much smaller, so they'll load faster.

7. **Consider the Sizes of Buttons and Icons:** When working with a mobile interface, you have a limited amount of space and some designers add too many buttons/icons. Another consideration is the size of the human fingertip. If the buttons/icons are too small, users could make errors with selecting the wrong one. Likewise, if there's not enough space between the buttons/icons, that can cause trouble as well. If in doubt, test your layouts and get feedback.

8. **Create a Core Application:** This means taking the most important features and building those into a core application experience. Additional functionality can be created by building plug-ins that can be purchased as necessary by the user. This avoids overloading the core part of the app with too many features.

9. **Create a Consistent Workflow:** This translates into making sure the user experience remains the same on all platforms. If you change that for each device, you'll confuse and annoy your users.

10. **Test the Design:** With any design, this is the most important aspect. If you've been following the strategies listed in this article you'll be testing your app every step of the way. Still, it's important to test the finished product and not only once but several times with different users. If there are problems, fix them, then test the result again.

TOPIC 3

15.3 Mobile platforms and architectures

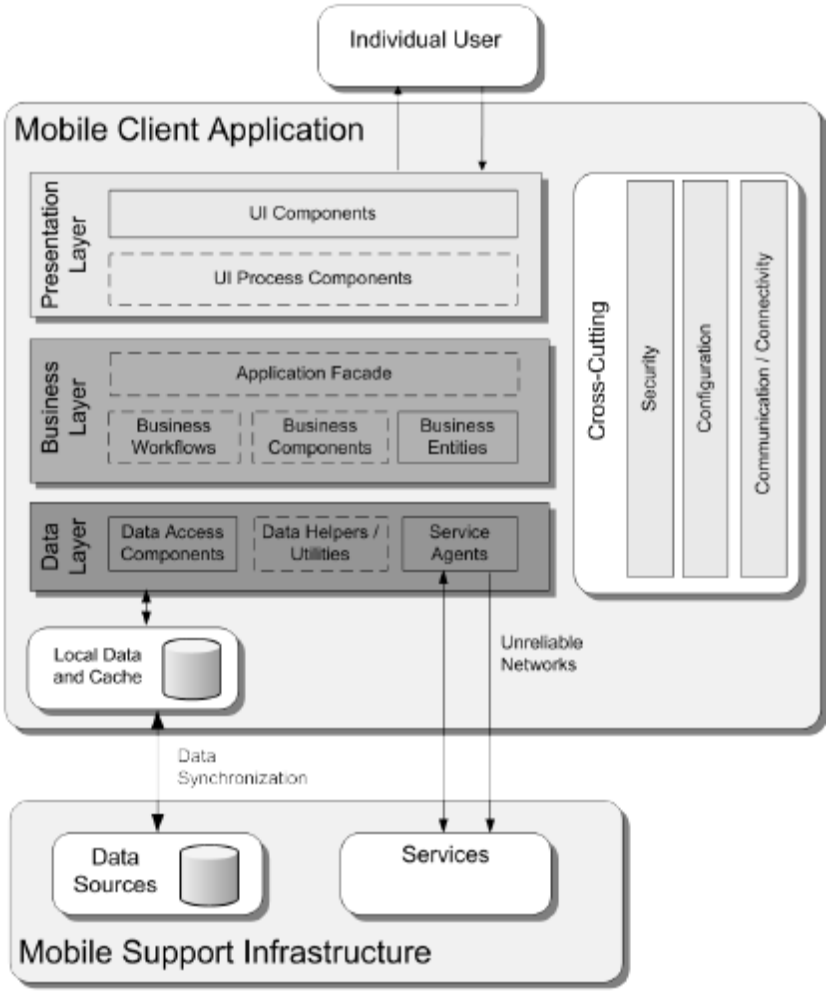


Figure 1 Common mobile application architecture

Architecture Frame

The following table lists the key areas to consider as you develop your architecture. Refer to the key issues in the table to understand where mistakes are most often made. The sections following this table provide guidelines for each of these areas.

Table 1 Architecture Frame

Area	Key issues
Authentication	• Lack of authorization across trust boundaries • Granular or improper authorization
Caching	• Caching volatile data not needed in an offline scenario • Caching sensitive data • Incorrect choice of caching store
Communication	• Incorrect choice of transport protocol • Chatty communication across physical and process boundaries • Failure to protect sensitive data
Concurrency and Transactions	• Not protecting concurrent access to static data • Deadlocks caused by improper locking • Not choosing the correct data concurrency model • Long-running transactions that hold locks on data • Using exclusive locks when not required
Configuration Management	• Lack of or incorrect configuration information • Not securing sensitive configuration information • Unrestricted access to configuration information
Coupling and Cohesion	• Incorrect grouping of functionality • No clear separation of concerns • Tight coupling across layers
Data Access	• Chatty calls to the database • Business logic mixed with data access code
Exception Management	• Failing to an unstable state • Revealing sensitive information to the end user • Using exceptions to control application flow • Not logging sufficient details about the exception
Layering	• Incorrect grouping of components within a layer • Not following layering and dependency rules
Logging and Instrumentation	• Lack of logging and instrumentation • Logging and instrumentation that is too fine-grained • Not making logging and instrumentation an option that is configurable at run time • Not suppressing and handling logging failures • Not logging business-critical functionality
State Management	• Using an incorrect state store • Not considering serialization requirements • Not persisting state when required
User Experience	• Not following published guidelines • Not considering accessibility

User Experience	• Not following published guidelines • Not considering accessibility • Creating overloaded interfaces with unrelated functionality
Validation	• Lack of validation across trust boundaries • Failure to validate for range, type, format, and length • Not reusing validation logic
Workflow	• Not considering management requirements • Choosing an incorrect workflow pattern • Not considering exception states and how to handle them

- ***Internet protocols for mobile applications***

Mobile IP is an Internet Engineering Task Force (IETF) standard communications protocol that is designed to allow mobile device users to move from one network to another while maintaining their permanent IP address. Defined in Request for Comments (RFC) 2002, Mobile IP is an enhancement of the Internet Protocol (IP) that adds mechanisms for forwarding Internet traffic to mobile devices (known as mobile nodes) when they are connecting through to other than their home network.

In traditional IP routing, IP addresses represent a topology. Routing mechanisms rely on the assumption that each network node will always have the same point of attachment to the Internet, and that each node's IP address identifies the network link where it is connected. Core Internet routers look at the IP address prefix, which identifies a device's network.
At the network level, routers look at the next few bits to identify the appropriate subnet. Finally, at the subnet level, routers look at the bits identifying a particular device. In this routing scheme, if you disconnect a mobile device from the Internet and want to reconnect through a different subnet, you have to configure the device with a new IP address, and the appropriate netmask and default router. Otherwise, routing protocols have no means of delivering packets because the device's IP address doesn't contain the necessary information about the current point of attachment to the Internet.

All the variations of Mobile IP assign each mobile node a permanent home address on its home network and a care-of address that identifies the current location of the device within a network and its subnets. Each time a user moves the device to a different network, it acquires a new care-of address. A mobility agent on the home network associates each permanent address with its care-of address. The mobile node sends the home agent a binding update each time it changes its care-of address using Internet Control Message Protocol (ICMP).
In Mobile IPv4, traffic for the mobile node is sent to the home network but is intercepted by the home agent and forwarded via tunneling mechanisms to the appropriate care-of address. Foreign agents on the visited network helps to forward datagrams. Mobile IPv6 was developed to minimize the necessity for tunneling and to include mechanisms that make foreign agents unnecessary.

Enhancements to the Mobile IP standard, such as Mobile IPv6 and Hierarchical Mobile IPv6 (HMIPv6), were developed to advance mobile communications by making the processes involved less cumbersome.

Further explanation

Definition of terms
Home network
The home network of a mobile device is the network within which the device receives its identifying IP address (home address).
Home address
The home address of a mobile device is the IP address assigned to the device within its home network.
Foreign network
A foreign network is the network in which a mobile node is operating when away from its home network.
Care-of address
The care-of address of a mobile device is the network-native IP address of the device when operating in a foreign network.
Home agent
A home agent is a router on a mobile node's home network which tunnels datagrams for delivery to the mobile node when it is away from home. It maintains current location (IP address) information for the mobile node. It is used with one or more foreign agents.
Foreign agent
A foreign agent is a router that stores information about mobile nodes visiting its network. Foreign agents also advertise care-of-addresses which are used by Mobile IP.
Binding
A binding is the association of the home address with a care-of address.

The Mobile IP allows for location-independent routing of IP datagrams on the Internet. Each mobile node is identified by its home address disregarding its current location in the Internet. While away from its home network, a mobile node is associated with a *care-of* address which identifies its current location and its home address is associated with the local endpoint of a tunnel to its *home agent*. Mobile IP specifies how a mobile node registers with its home agent and how the home agent routes datagrams to the mobile node through the *tunnel*....

Applications
In many applications (e.g., VPN, VoIP), sudden changes in network connectivity and IP address can cause problems. Mobile IP was designed to support seamless and continuous Internet connectivity.

Mobile IP is most often found in wired and wireless environments where users need to carry their mobile devices across multiple LAN subnets. Examples of use are in roaming between overlapping wireless systems, e.g., IP over DVB, WLAN, WiMAX and BWA.

Mobile IP is not required within cellular systems such as 3G, to provide transparency when Internet users migrate between cellular towers, since these systems provide their own data link layer handover and roaming mechanisms. However, it is often used in 3G systems to allow seamless IP mobility between different packet data serving node (PDSN) domains.

Operational principles

The goal of IP Mobility is to maintain the TCP connection between a mobile host and a static host while reducing the effects of location changes while the mobile host is moving around, without having to change the underlying TCP/IP. To solve the problem, the RFC allows for a kind of proxy agent to act as a *middle-man* between a mobile host and a correspondent host.
A mobile node has two addresses – a permanent home address and a care-of address (CoA), which is associated with the network the mobile node is visiting. Two kinds of entities comprise a Mobile IP implementation:

- A ***home agent*** (HA) stores information about mobile nodes whose permanent home address is in the home agent's network. The HA acts as a router on a mobile host's (MH) home network which tunnels datagrams for delivery to the MH when it is away from home, maintains a location directory (LD) for the MH.
- A ***foreign agent*** (FA) stores information about mobile nodes visiting its network. Foreign agents also advertise care-of addresses, which are used by Mobile IP. If there is no foreign agent in the host network, the mobile device has to take care of getting an address and advertising that address by its own means. The FA acts as a router on a MH's visited network which provides routing services to the MH while registered. FA detunnels and delivers datagrams to the MH that were tunneled by the MH's HA
  The so-called *Care of Address* is a termination point of a tunnel toward a MH, for datagrams forwarded to the MH while it is away from home.
- **Foreign agent care-of address**: the address of a foreign agent that MH registers with co-located care-of address: an externally obtained local address that a MH gets.
  Mobile Nodes (MN) are responsible for discovering whether it is connected to its home network or has moved to a foreign network. HA's and FA's broadcast their presence on each network to which they are attached. They are not solely *responsible* for discovery, they only play a part. RFC 2002 specified that MN use agent discovery to locate these entities. When connected to a foreign network, a MN has to determine the foreign agent care-of-address being offered by each foreign agent on the network.

A node wanting to communicate with the mobile node uses the permanent home address of the mobile node as the destination address to send packets to. Because the home address logically belongs to the network associated with the home agent, normal IP routing mechanisms forward these packets to the home agent. Instead of forwarding these packets to a destination that is physically in the same network as the home agent, the home agent redirects these packets towards the remote address through an IP tunnel by encapsulating the datagram with a new IP header using the care of address of the mobile node.

When acting as transmitter, a mobile node sends packets directly to the other communicating node, without sending the packets through the home agent, using its permanent home address as the source address for the IP packets. This is known as triangular routing or "route optimization" (RO) mode. If needed, the foreign agent could employ *reverse tunneling* by tunneling the mobile node's packets to the home agent, which in turn forwards them to the communicating node. This is needed in networks whose gateway routers check that the source IP address of the mobile host belongs to their subnet or discard the packet otherwise. In Mobile IPv6 (MIPv6), "reverse tunneling" is the default behaviour, with RO being an optional behaviour.

In scenarios when both sides of communication are mobile nodes, communicating via Mobile IP solutions adds additional overhead that decreases efficient packet payloads. As a solution, in 2012 researchers developed a method to decrease the size of overhead in situations, so that more payloads can be transferred in each IP packet in the discussed scenarios. In the proposed method, the tunnel manager is changed to act as a DNS, so that sending MN addresses are no longer required.

Performance

A performance evaluation of Mobile IPv6, carried out by NEC Europe, can be found at the ACM Digital Library, under the entry "A simulation study on the performance of mobile IPv6 in a WLAN-based cellular network", from the Elsevier *Computer Networks Journa*l (CNJ), special issue on The New Internet Architecture, September 2002.

Additionally, a performance comparison between Mobile IPv6 and some of its proposed enhancements (Hierarchical Mobile IPv6, Fast Handovers for Mobile IPv6 and their Combination) is available under the entry "A performance comparison of Mobile IPv6, Hierarchical Mobile IPv6, fast handovers for Mobile IPv6 and their combination", from the ACM SIGMOBILE *Mobile Computing and Communications Review* (MC2R), Volume 7, Issue 4, October, 2003.

Development

Enhancements to the Mobile IP technique, such as Mobile IPv6 and Hierarchical Mobile IPv6 (HMIPv6) defined in RFC 5380, are being developed to improve mobile communications in certain circumstances by making the processes more secure and more efficient. HMIPv6 explanation can be found at Hierarchical-Mobile-IPv6.

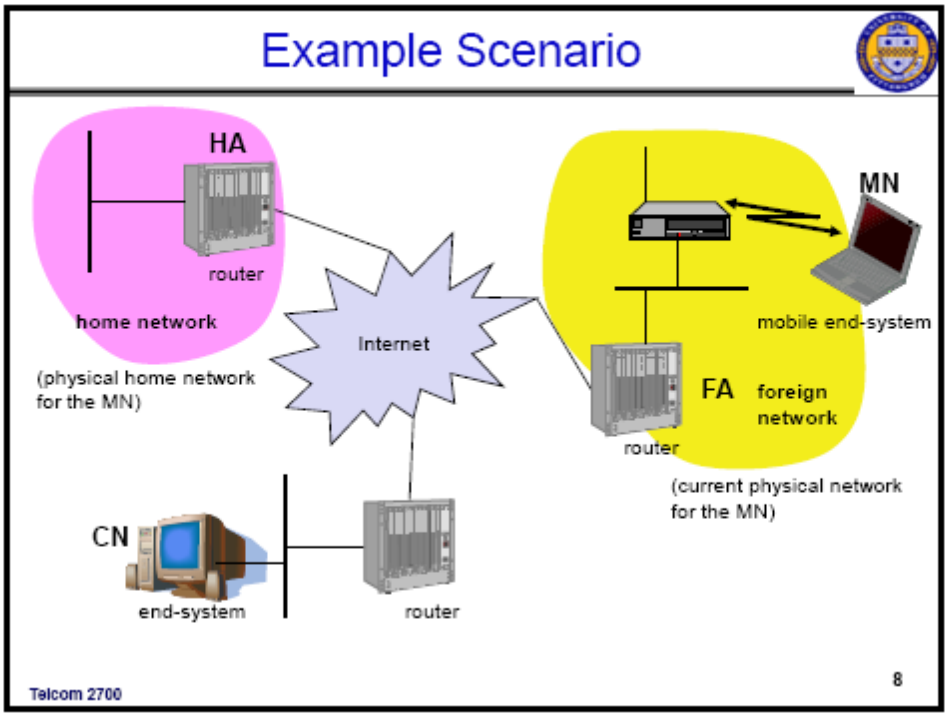
Researchers create support for mobile networking without requiring any pre-deployed infrastructure as it currently is required by MIP. One such example is Interactive Protocol for Mobile Networking (IPMN) which promises supporting mobility on a regular IP network just from the network edges by intelligent signaling between IP at end-points and application layer module with improved quality of service.

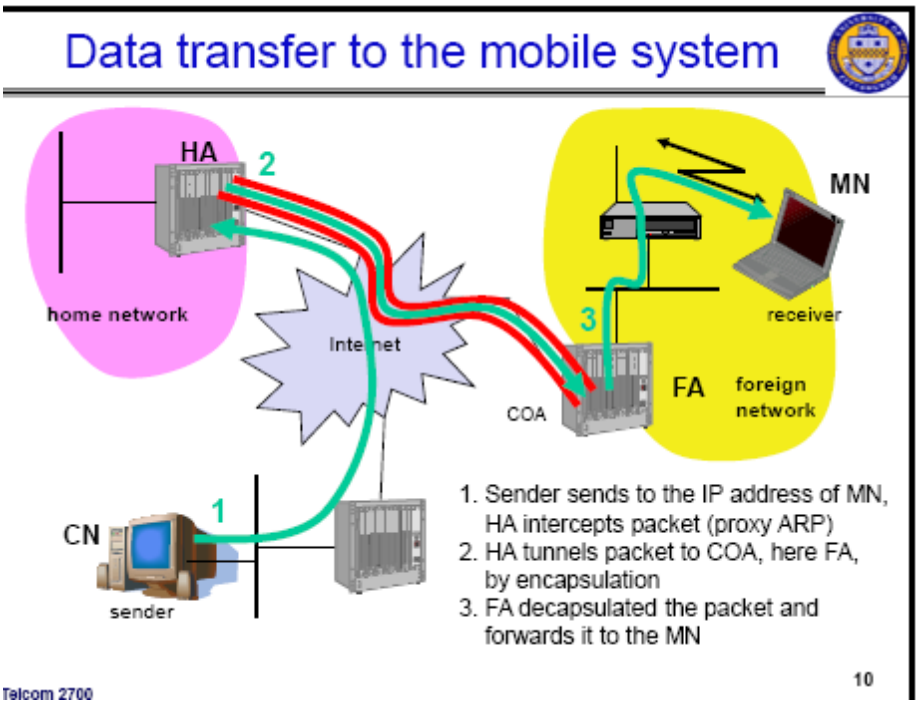
Researchers are also working to create support for mobile networking between entire subnets with support from Mobile IPv6. One such example is Network Mobility (NEMO) Network

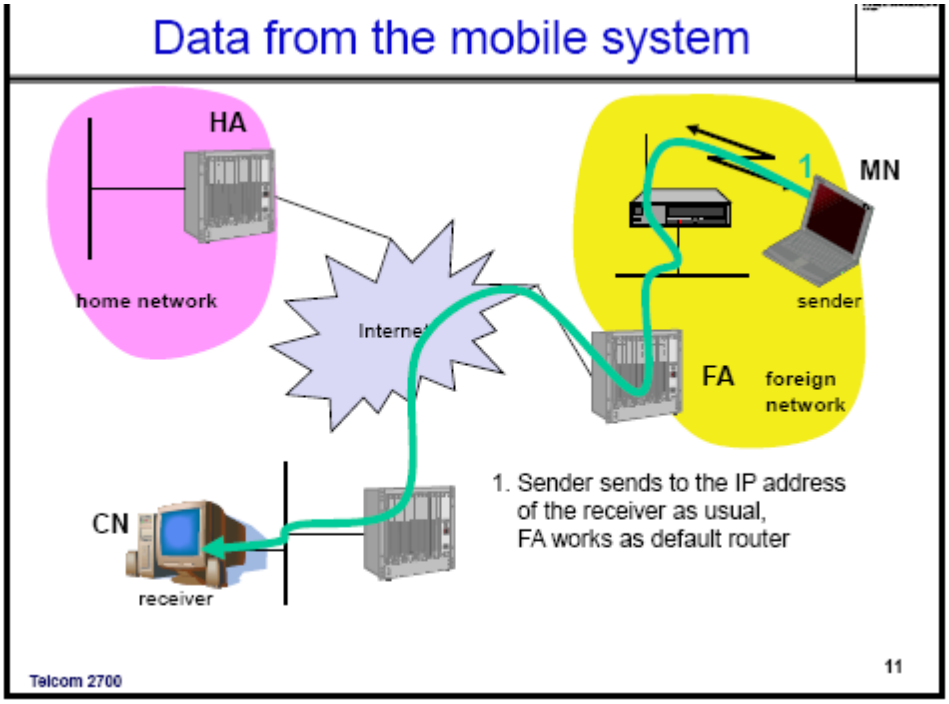
Mobility Basic Support Protocol by the IETF Network Mobility Working Group which supports mobility for entire Mobile Networks that move and to attach to different points in the Internet. The protocol is an extension of Mobile IPv6 and allows session continuity for every node in the Mobile Network as the network moves.

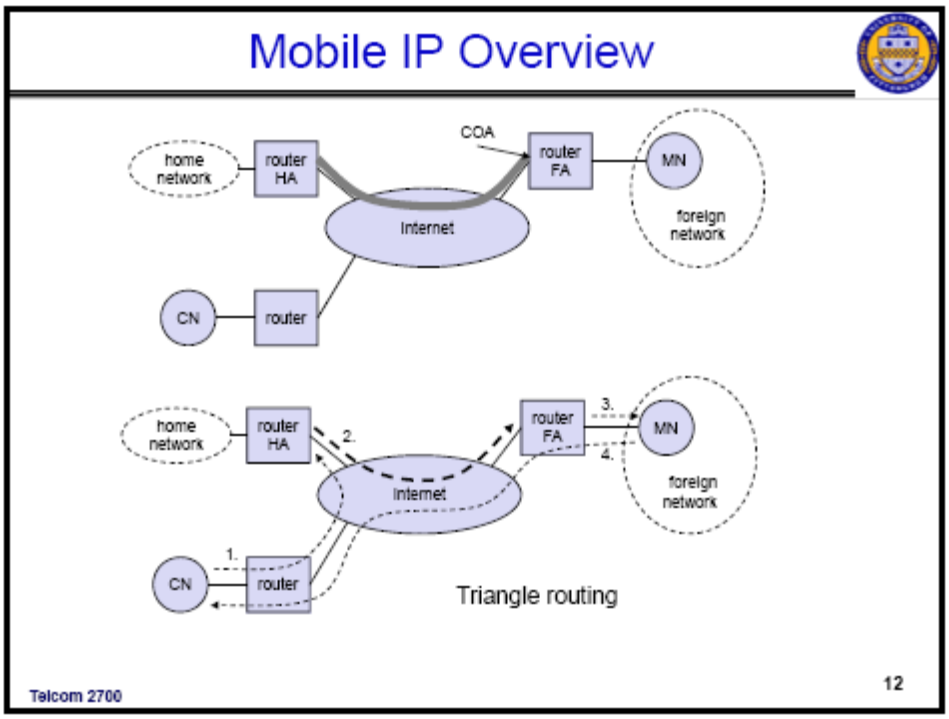
Changes in IPv6 for Mobile IPv6

- A set of mobility options to include in mobility messages
- A new Home Address option for the Destination Options header
- A new Type 2 Routing header
- New Internet Control Message Protocol for IPv6 (ICMPv6) messages to discover the set of home agents and to obtain the prefix of the home link
- Changes to router discovery messages and options and additional Neighbor Discovery options
- Foreign Agents are no longer needed









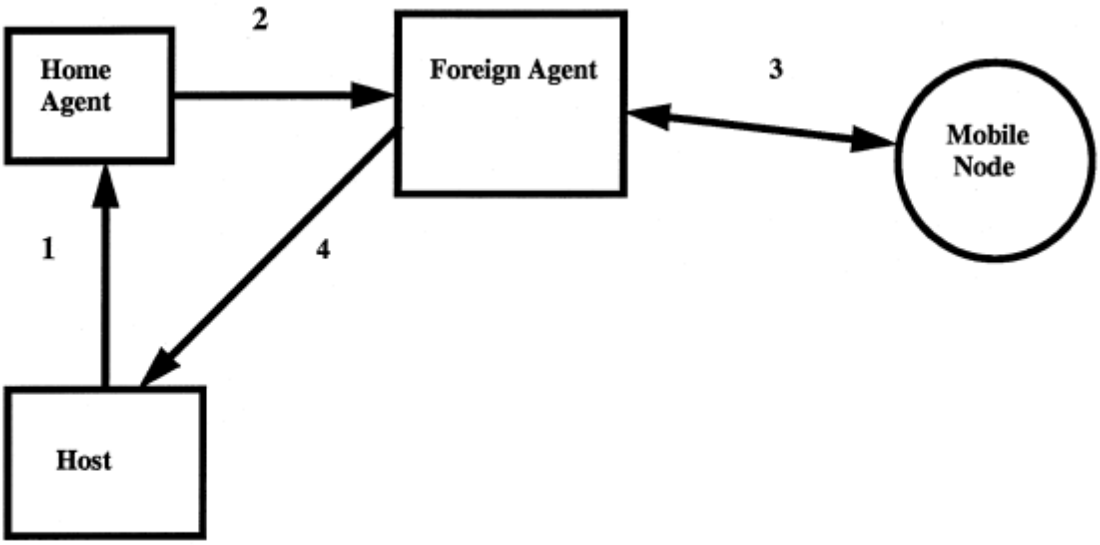


Figure 25.1 Transmission of messages in a mobile IP environment.

- ***Mobile application distribution platforms and environments***

Distribution

The two biggest app stores are Google Play for Android and App Store for iOS.

THIS IS A FREE SAMPLE OF THE ACTUAL NOTES

TO GET FULL/COMPLETE NOTES:

Call/Text/Whatsapp:0713440925

You can also write to us at: topexamskenya@gmail.com
or info@mykasnebnotes.com

To download more resources visit: http://www.mykasnebnotes.com

For updates and insights like us on Facebook