

Ben Dodson

4/14/11

Rule your digital world with the mobile Hot Potato



NFC (Near Field Communication)

- NFC is a wireless technology for interacting with devices with a touch.
 - ~ 5cm range, ~424 kbit/s bandwidth.
- Supports Active and Passive devices
 - Active: powered devices; phones, PC readers, etc
 - Passive: unpowered. Stickers, ID badges, credit card.

Interactions at a Touch

- Why do people care about NFC?
 - “It’s coming.” Payments, coupons, ticketing.
 - Finally reaching the latest generation of mobile devices.
- Why do *we* care about NFC?
 - Is short-ranged, intention-oriented, quick setup.
 - Bidirectional, symmetric communication.
 - NFC is low-power, can be always on.
 - Lives “behind the screen”, in the background.

We like our phones.

- Phones
 - Powerful.
 - Connected.
 - Always with us.
 - Personal.
- Phones have a sense of **identity**.



We like our other devices, too.



NFC helps us bring them together

- Phone is your digital personality, which it brings to other devices.



What is Hot Potato?

- Instant transfer of the one thing in my hand
- Instant trigger of the appropriate reaction upon receipt of the thing



Hot Potato is useful (demos)

- Phone-to-Phone, Phone-to-PC, Phone-to-TV
- Sharing links
- Sharing files
- Running applications

Hot Potato Paradigm

- Foreground application “owns” the radio.
- Designate the “hot potato” in advance
 - `mNfc.share(object);`
- Specify the handler
 - Foreground app: `mNfc.addNdefHandler(handler)`
 - Otherwise: Intent filters, protocol handlers
- NFC touch issues bidirectional exchange

Context-Aware NFC

- One touch, many interactions
 - Phone-to-Phone
 - Two people exchanging data.
 - Links, files, contact information, applications
 - Phone-to-PC
 - User-to-self interactions
 - Authenticated links, personal utilities, files, payments
 - Phone-to-TV
 - User-to-device
 - Multimedia, applications
 - Handle content with no further action

Challenges of Sharing with NFC

1. What if the device doesn't have NFC?
2. How do we share more than just data?
 - Cross-platform applications
3. How to keep an ongoing interaction?
 - Multi-partied, real-time communication

Key Concept: Virtual NFC

- Standardize on NDEF (language of NFC)
- Standardize on single-packet exchange
- Implement across devices without NFC

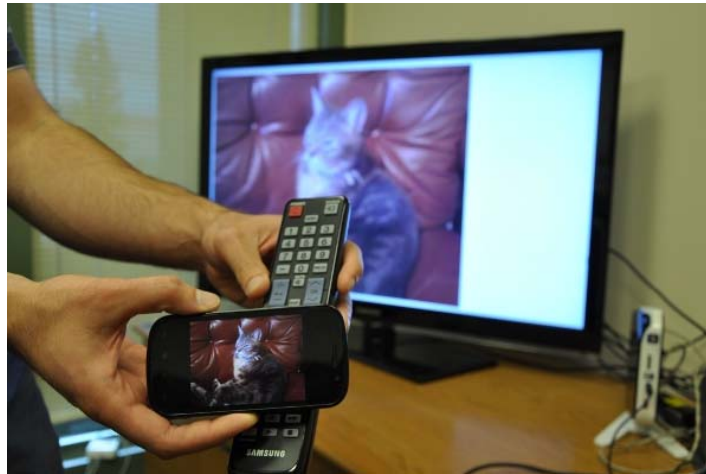
NDEF: The language of NFC

- NFC Forum defines a data format called NDEF (NFC Data Exchange Format)
- NDEF is at the same level as HTTP, OBEX
 - Can be run over any link layer
 - But is designed for use with NFC
- NDEF Exchange is not request/response
 - Especially when dealing with passive NFC devices.
 - Requires a packet structure that contextualizes its content.

NDEF: The language of NFC

- NDEF Format
 - NDEF Message has 1 or more NDEF Records.
 - NDEF Record has:
 - Type Name Format (TNF)
 - URI, Mime-Type, Well-Known-Type, External
 - RTD (Record Type Format)
 - "uri", "application/jpeg", "connection handover" ...
 - And a payload of arbitrary bytes.

1. Devices without NFC

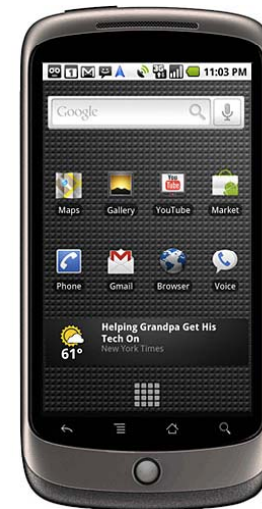


Connection Handover

- Connection Handover is a specially formatted NDEF message
 - Generic framework defined by NFC Forum
 - Negotiated handover, or static handover
- libhotpotato recognizes “NDEF Exchange” handover
 - Runs bidirectional NDEF exchange protocol over bluetooth or tcp
 - Invisible to developer.

Preparing a non-NFC device

o. Associate NFC tag with a “touchless” phone



Preparing a non-NFC device

1. Run Hot Potato daemon.

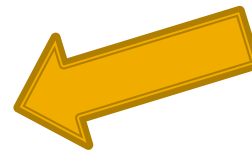


QR code encodes
Bluetooth address and public key.



Preparing a non-NFC device

2. A friend with NFC helps out

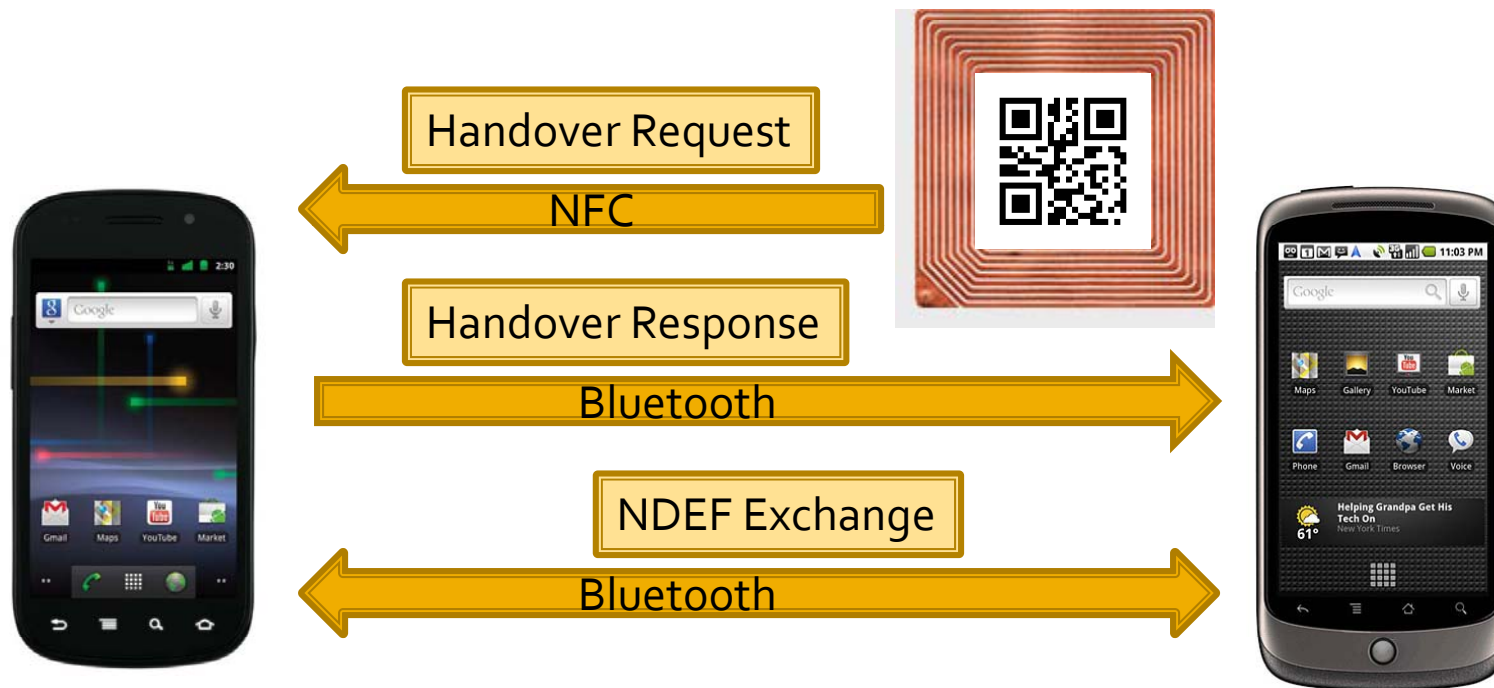


Preparing a non-NFC device

3. Writes request to tag



Connection Handover



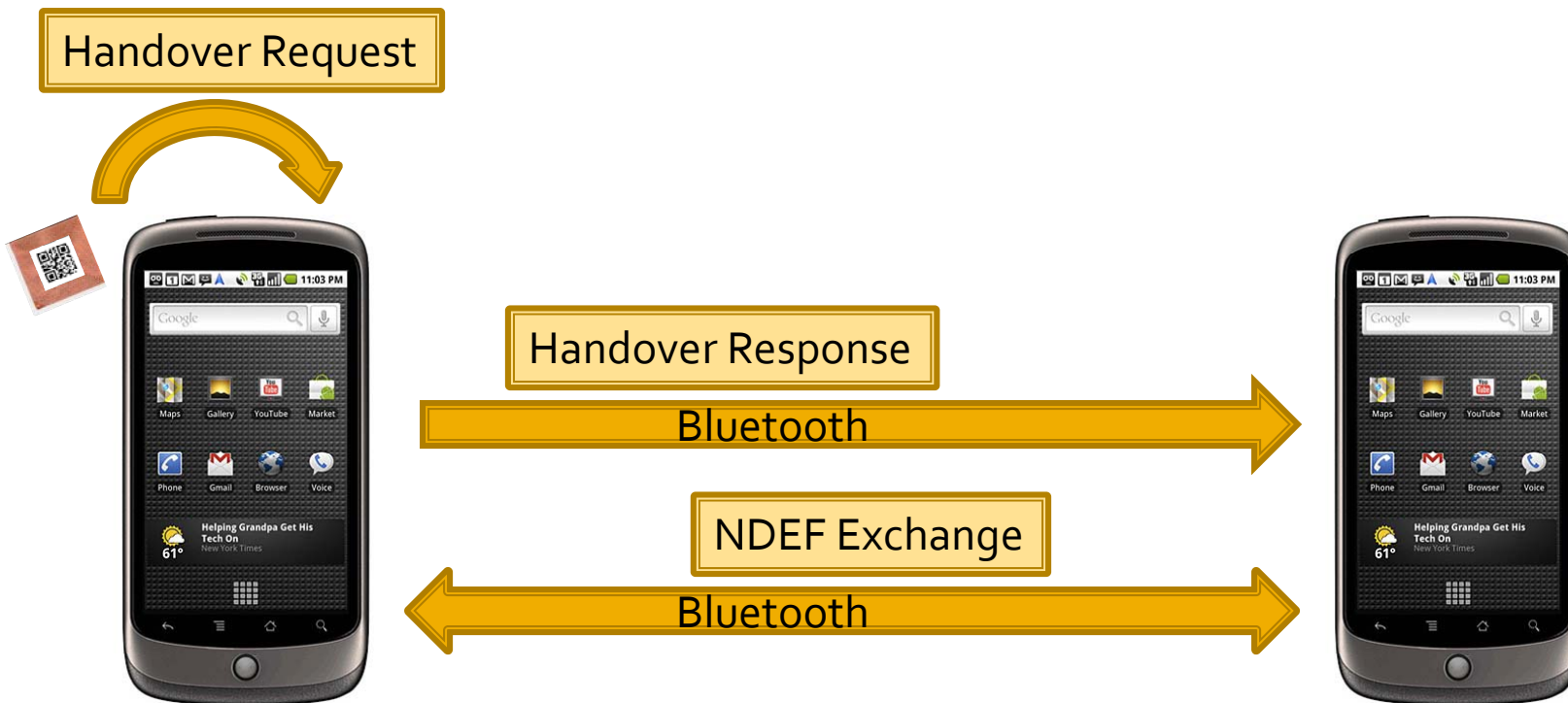
- All hidden from the developer

NDEF Exchange as First Class Object

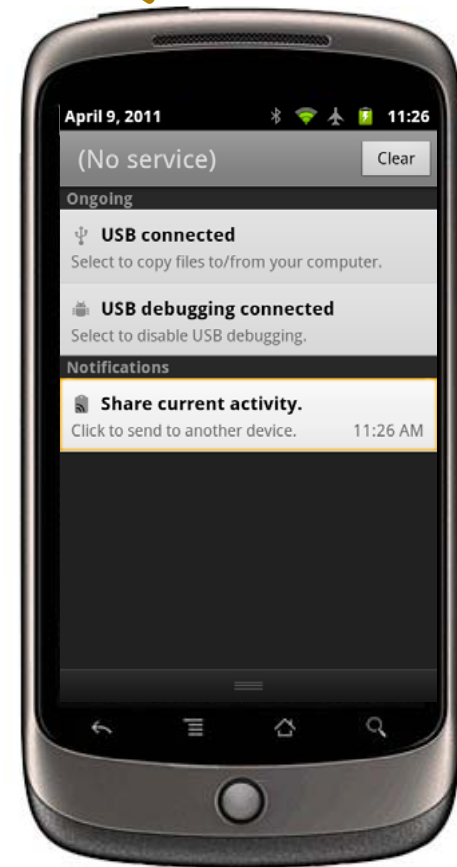
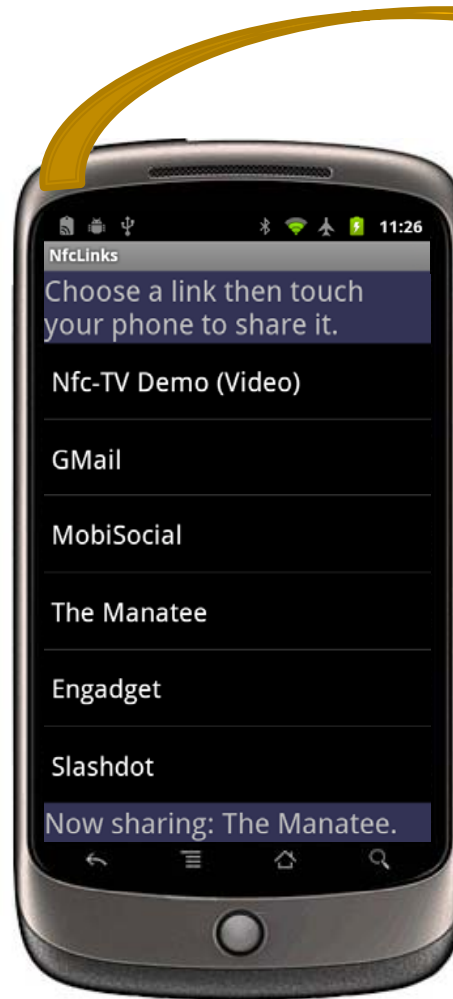
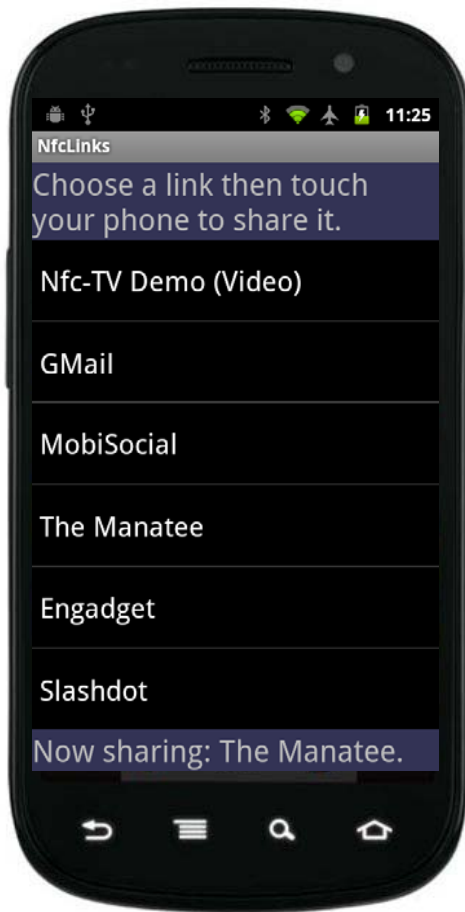
- An NDEF Exchange request is representable as a data structure
- Can pass it between devices, store it for later
- Virtualized NFC: Can use the NFC interaction without the NFC radio



Virtual NFC (Ndef Exchange)



Virtual NFC



2. Sharing applications

- Application shares itself over NDEF Exchange
 - May be received by
 - Same application
 - Different application
 - Some Platform
- How do we represent an application so it can be understood in any context?

Application Manifest for Cross-Platform Apps

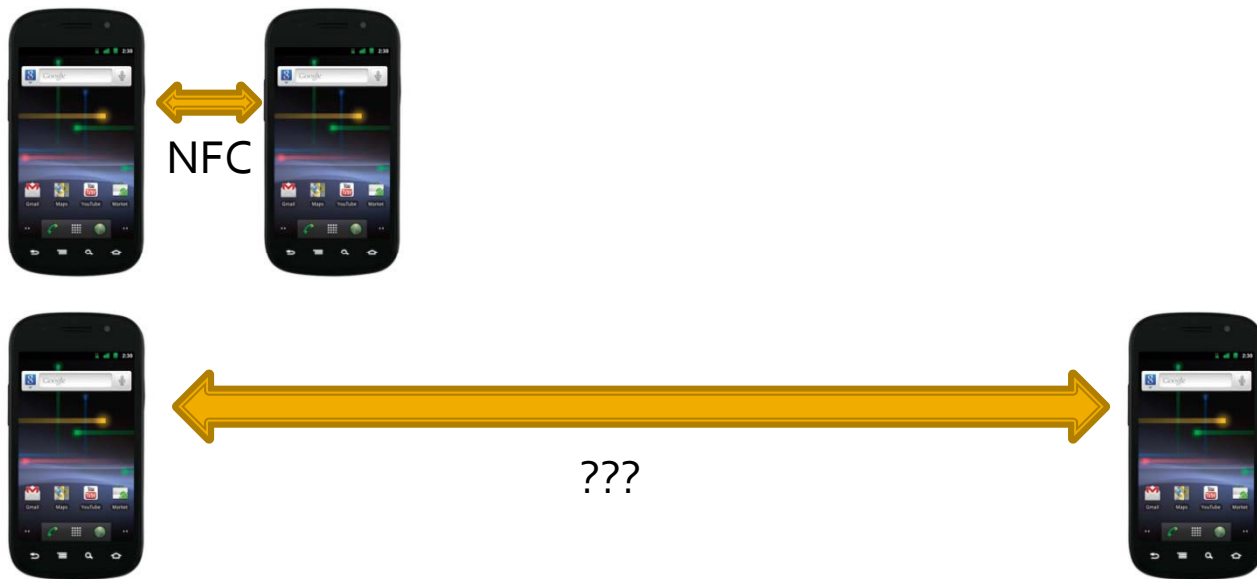
Application Manifest for Cross-Platform Apps

- Encodes a list of:
 - (platform, app-reference, app-argument)
 - Whiteboard:

Platform	App reference	App argument
Android+Market	mobisocial.whiteboard	junction://sb/m48727f
iPhone+Store	mobisocial.whiteboard	whiteboard:// junction://sb/m48727f
Web	mobisocial.stanford.edu /whiteboard	{jxuri : junction://sb/m48727f}

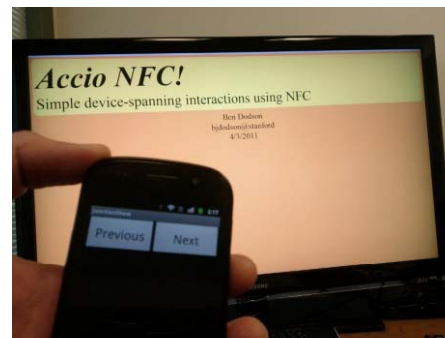
3. On-Going Interaction

- If NDEF exchange is a single-packet transaction, how do we have an ongoing interaction?



Junction

- Cross-platform framework for device-spanning applications.
 - Provides real-time communication across devices
 - Running session represented as URI, separate from code URL.



Why not Bump?

- Biggest differences between NFC and Bump
 - Requires user interaction to bring up
 - Blocks current activity
 - Receiver must launch special program
 - All data passes through Bump cloud service
- NFC is always on
- Lives behind the screen



What can we do?

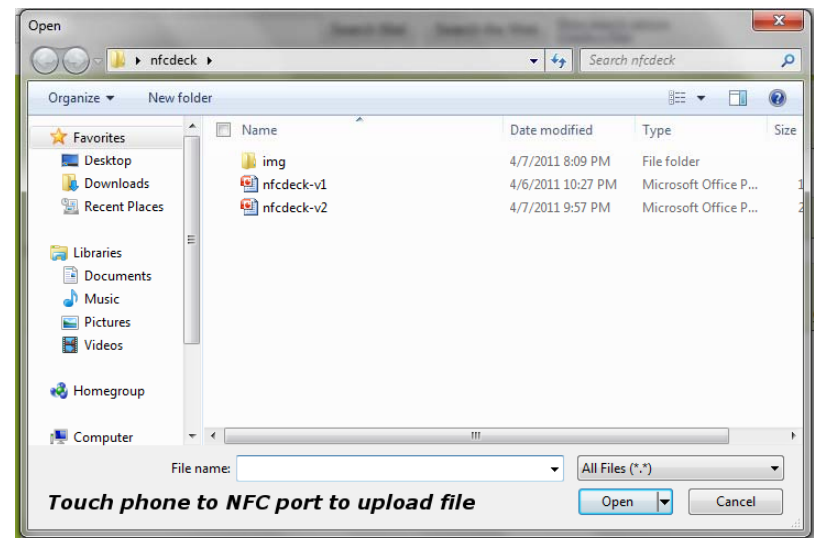
- NFC Forum specifies radio and “firmware”
 - Large consortium good for base standards, not suited for rapid software evolution
- Platform vendors need higher level standards
 - NDEF Exchange protocol
 - Handover protocols
 - App Manifest

Conclusions

- NFC enables o-click interactions
 - What's in front of the screen represented behind the screen
- “NDEF exchange” as interesting as NFC
 - NFC as a *system*, not just a radio.
 - Open standard around this exchange.
- Right abstraction lets us extend the experience of NFC for users, without burdening developers
- Hot Potato and libhotpotato available now
 - <http://github.com/mobisocial>
- Thanks!

Looking Forward

- Nfc.js
 - Allow web pages to interact with NFC directly
- OAUTH2-based mobile bookmarks
- Better desktop integration
- Handover environments



Establish a Bluetooth Connection

```
onBluetoothConnected  
    (BluetoothSocket s) {  
  
    // Your code here  
  
};
```

Roadmap

- Observations on our digital world
- Augmenting interactions with NFC
- Defining the Hot Potato
- Context-Sensitive Interactions with NFC
- Virtual NFC
- Conclusions

Contributions

- Vision: Phones makes our digital experience coherent
 - Centralized management of identities & personal assets.
 - Hot Potato connects them to friends and surrounding physical devices
- Hot Potato abstraction: user and application layer
 - Uniform experience of sharing with an NFC tap.
 - Virtual NFC: NDEF Exchange Connection Handover
 - For transferring large data types
 - Inter-operability across devices without NFC
 - Compatibility across platforms
 - Apple unlikely to adopt “Android NDEF Push Protocol”
- Open-source Android library → standard across platforms?

Distributed Applications

- Require three problems be solved:
 - **Naming.** How to refer to another device
 - **Communication.** How to send messages
 - **Contextualization.** How to interpret messages
- NFC solves all three with a tap.

What is the Hot Potato?

- Model: Two devices touch, “want” to interact
 - Intentional Initiations
 - Focus on one object at a time
- Exploit NFC’s o-click potential
 - Enable *Spontaneous Interactions*
- Contextually-aware message exchange

Enabling the Digital Hot Potato

- Question: How do you represent what's on screen to a random visitor?



Programming Paradigm

- Two devices touch triggering an interaction.
 - Simultaneous event between two devices
 - Developer prepares data in advance.
 - `makeDataAvailable(...)`, not `sendDataNow(...)`
 - `onDataReceived(...)`, not `data = waitForData(...)`



NDEF: Why it Matters

- NDEF defines the “hot potato”
 - Self-contained, well-typed data structure
 - Contain contextual invocation information
- NDEF Exchange
 - Make a single object available, provide as much context as possible up front.

What is Hot Potato?

NFC-inspired user interaction

- Based on the technique we call virtual NFC

With one tap

- Activity you are working with is shared (hot potato)
 - URL, text
 - Large data types: Photos, videos, data streams
 - Multi-party games being played
- The sharing device launches the right operation
 - Devices include phones
 - PCs, TVs with no built-in NFC

Context-Sensitive Interactions: Phone-to-Phone

- *Share my identity with others*
- Phones will have active NFC radios
- Phones are always associated with people
 - *Two phones touch, two people interact.*

Context-Sensitive Interactions: Phone-to-PC

- *Use my identity across devices*
- Typically, PC and Phone are owned by the same user.
 - *User-to-self interaction*
- Authentication, bookmarks, settings, payments, ...

Context-Sensitive Interactions: Phone-to-TV

- *My digital assets to any device.*
- TVs are typically lacking in input devices.
 - *User-to-device interaction*
 - *Couch Potato*
- NFC lets us bring our phone personalization to our TV.
 - Our photos, music, videos with a touch
 - Our apps

Why not QR?

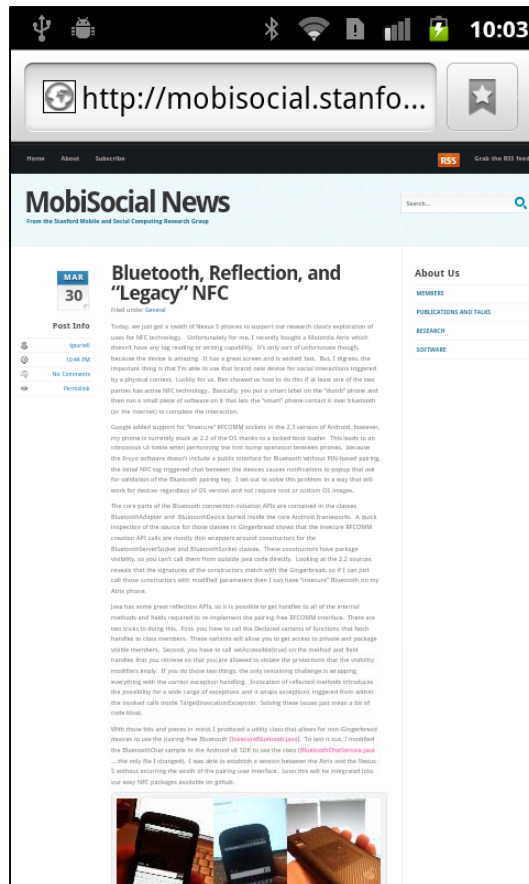
- Biggest differences between NFC and QR
 - QR code must be displayed on screen
 - Requires user interaction to bring up
 - Blocks current activity
 - Receiver must launch special program

- NFC is always on
- Lives behind the screen



NFC is for Sharing!

0



The screenshot shows a mobile browser interface with a status bar at the top displaying various icons and the time 10:03. The address bar shows the URL <http://mobisocial.stanfo...>. Below the address bar is a navigation bar with links for Home, About, and Subscribe, along with an RSS icon and a 'Grab the RSS feed' link. The main content area features the 'MobiSocial News' logo and a search bar. The article title is 'Bluetooth, Reflection, and "Legacy" NFC', dated March 30. The article text discusses the author's research on NFC technology, mentioning a Motorola Atrix phone and the challenges of using NFC on a locked boot loader. It details the author's discovery of the Bluetooth connection classes in the Gingerbread source code and the use of reflection to access internal methods. The article concludes with a note about the author's GitHub repository for NFC packages.

Bluetooth, Reflection, and "Legacy" NFC

Read this online


Today, we just got a batch of Nexus S phones to support our research client's exploration of uses for NFC technology. Unfortunately for me, I recently bought a Motorola Atrix which doesn't have any tag reading or writing capability. It's only use of NFC is for pairing, because the device is amazing. It has a great screen and a locked boot. But, I agree, the important thing is that I'm able to use that brand new device for social interactions triggered by a physical context. Lucky for us, Ben showed us how to do this if at least one of the two parties has active NFC technology. Basically, you get a small label on the "victim" phone and then run a small piece of software on it that lets the "victim" phone connect to over Bluetooth (or the internet) to complete the interaction.

Google added support for "insecure" BECOMM sockets in the 2.3 version of Android, however, my phone is currently stuck at 2.2 so the OS thinks it's a locked boot loader. This leads to an interesting UI issue when performing the time stamp operation between phones. Because the fringe software doesn't include a public interface for Bluetooth without PIN-based pairing, the initial NFC tag triggered that between the devices causes mechanisms to prompt that ask for validation of the Bluetooth pairing key. I set out to solve this problem in a way that will work for devices regardless of OS version and not require root or custom OS images.

The core parts of the Bluetooth connection-related APIs are contained in the classes BluetoothAdapter and BluetoothDevice based inside the core Android Frameworks. A quick inspection of the source for these classes in Gingerbread shows that the insecure BECOMM creation APIs are mostly thin wrappers around constructors for the BluetoothDevice and BluetoothAdapter classes. These constructors have package visibility, so you can't call them from outside Java code directly. Looking at the 2.2 sources reveals that the signatures of the constructors match with the Gingerbread, so if I can just call those constructors with modified parameters then I can have "insecure" Bluetooth on my Atrix phone.

Java has some great reflection APIs, so it is possible to get handles to all of the internal methods and fields required for re-implementing the pairing-free BECOMM interface. There are two tricks to doing this. First, you have to call the declared arrays of methods and field handles to class members. These variants will allow you to get access to private and package visible members. Second, you have to call `setAccessible(true)` on the method and field handles that you intend to use. This is allowed to violate the protections that the mobility modifiers bring. If you do these two things, the only remaining challenge is wrapping everything with the correct exception handling. Inspection of relevant methods introduces the possibility for a wide range of exceptions, and it wasn't surprising triggered from within the Android calls inside `TargetInvocationException`. Solving these issues just means a bit of code fiddling.

With these bits and pieces in mind, I prototyped a utility class that allows for non-Gingerbread devices to use the pairing-free Bluetooth (`BluetoothWithoutPin.java`). To test it out, I modified the BluetoothChat sample in the Android v2.2 SDK to use the class `BluetoothWithoutPin.java`... and only that (changed). I was able to establish a session between the Atrix and the Nexus S, without requiring the touch of the pairing user interface. Since this will be integrated into our own NFC packages available on github.



NFC is for Sharing!

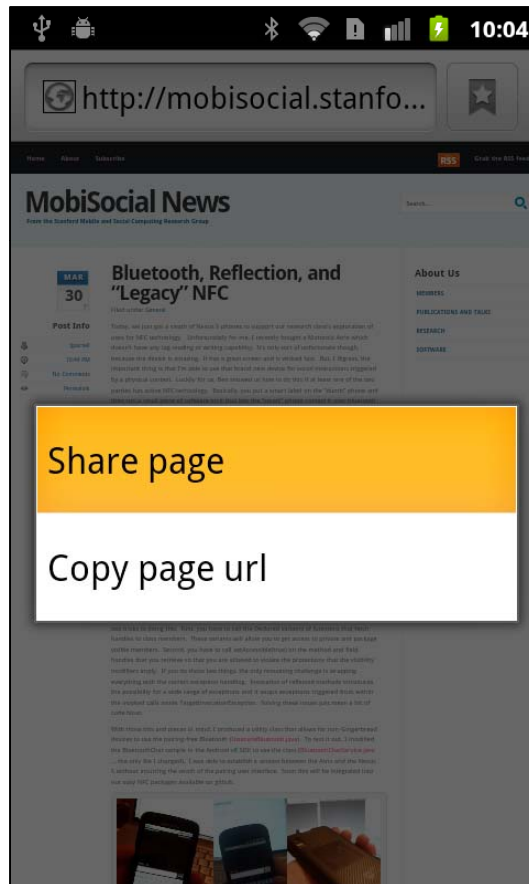
1



The screenshot shows a mobile browser interface. At the top, there's a status bar with icons for USB, Android, Bluetooth, Wi-Fi, signal strength, battery, and the time 10:04. Below that is a browser address bar showing the URL <http://mobisocial.stanfo...> and a star icon for bookmarks. The page header includes 'Home About Subscribe' and an 'RSS' icon with the text 'Grab the RSS feed'. The main content area features the 'MobiSocial News' logo and a search bar. The article title is 'Bluetooth, Reflection, and "Legacy" NFC' with a date of 'MAR 30' and a 'Read this article' link. The 'Post Info' section shows 'Start', 'Total: 0', 'No Comments', and 'Permalink'. The article text discusses NFC technology, mentioning a Motorola Atrix phone and the challenges of NFC on Android. It details the use of reflection APIs and the need for specific permissions like `android.permission.NFC` and `android.permission.NFC_AWARE`. The article concludes with a note about the author's GitHub repository for NFC packages.

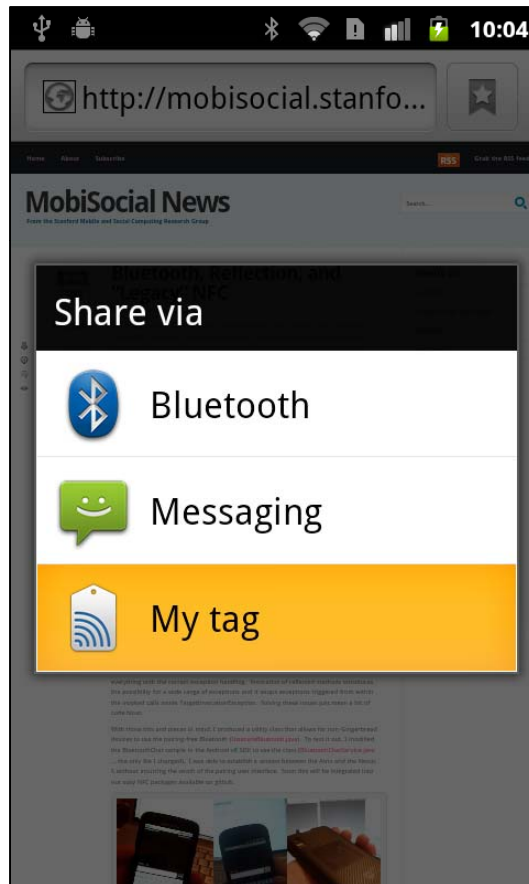
NFC is for Sharing!

2



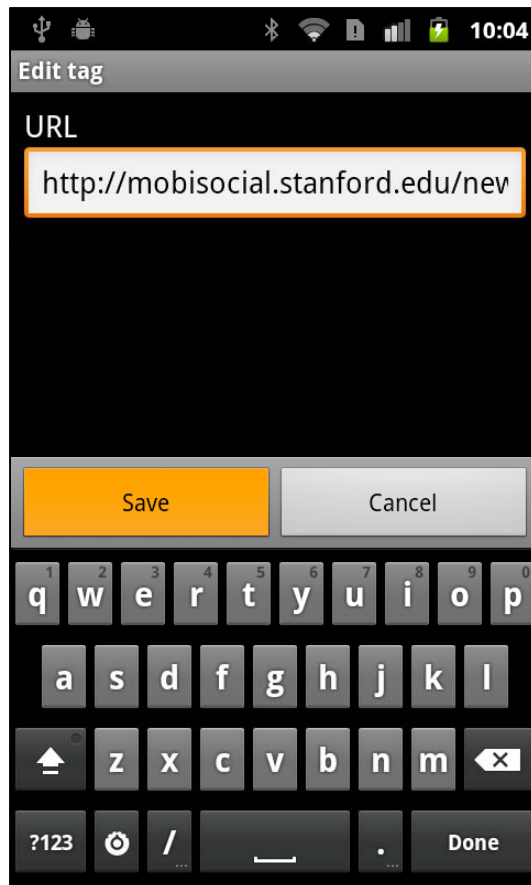
NFC is for Sharing!

3



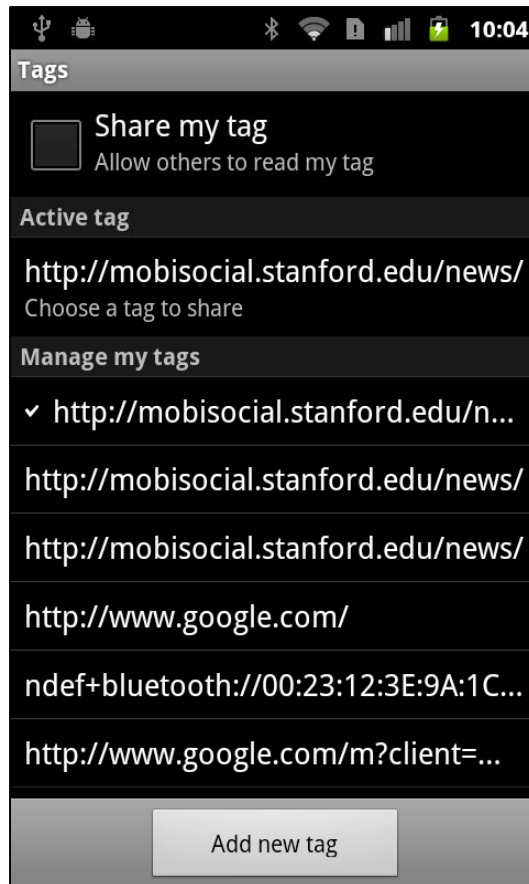
NFC is for Sharing!

4



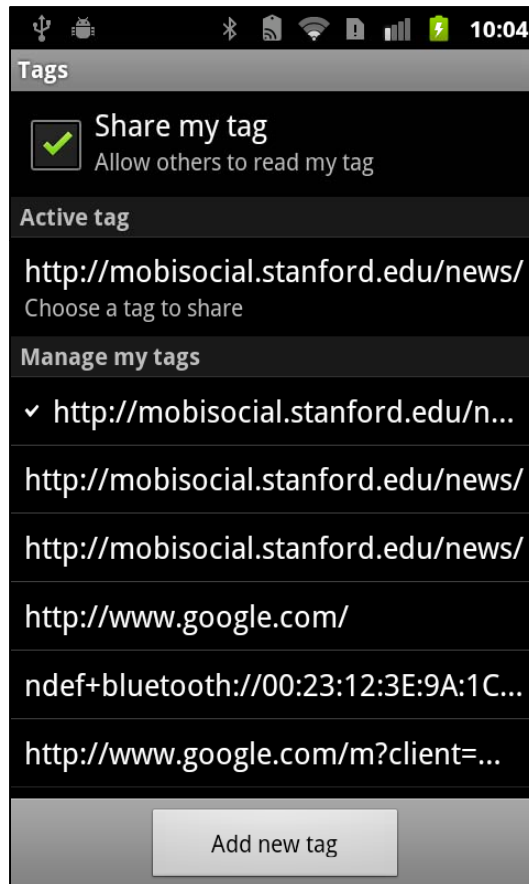
NFC is for Sharing!

4



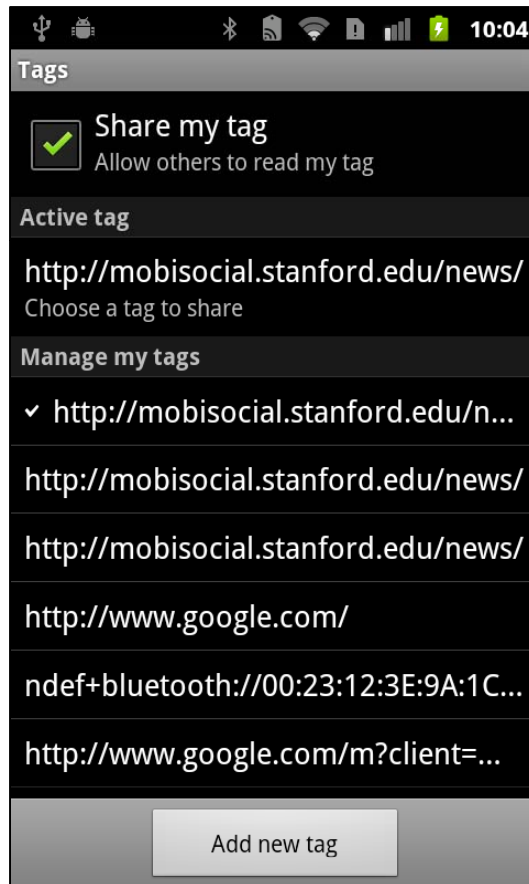
NFC is for Sharing!

5



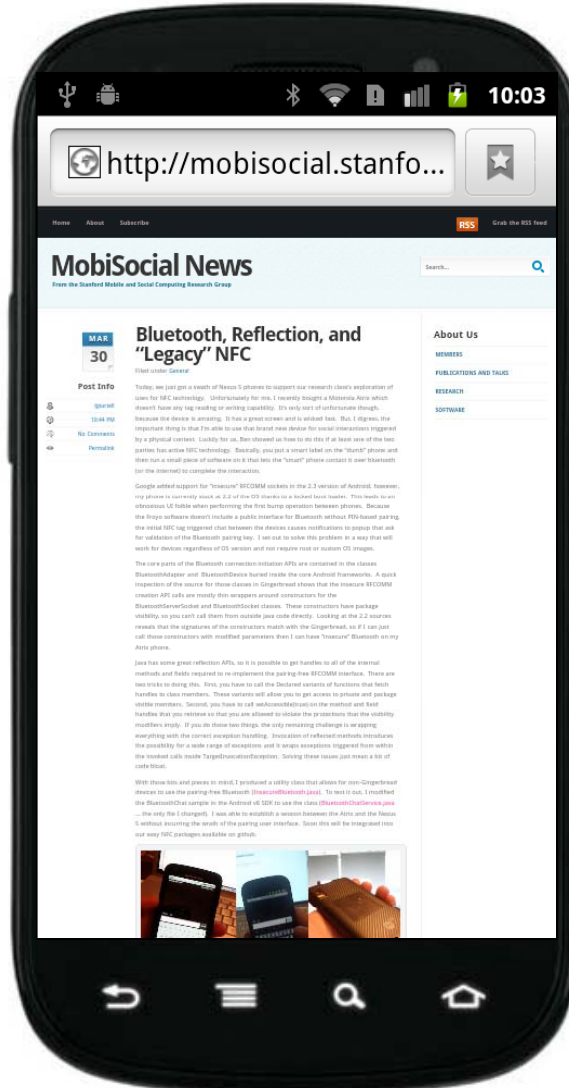
NFC is for Sharing!

5



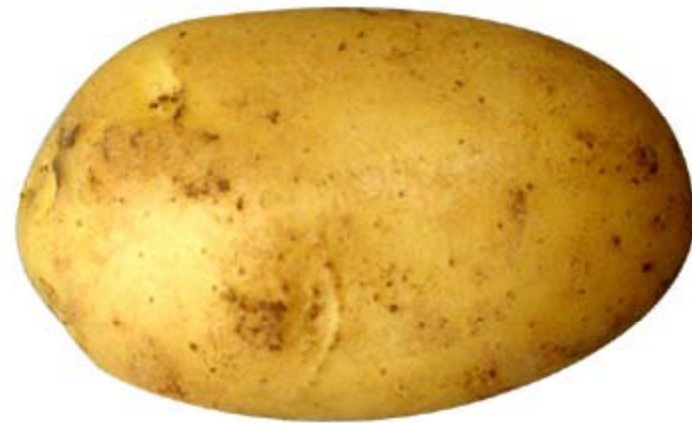
NFC is for Sharing?

5



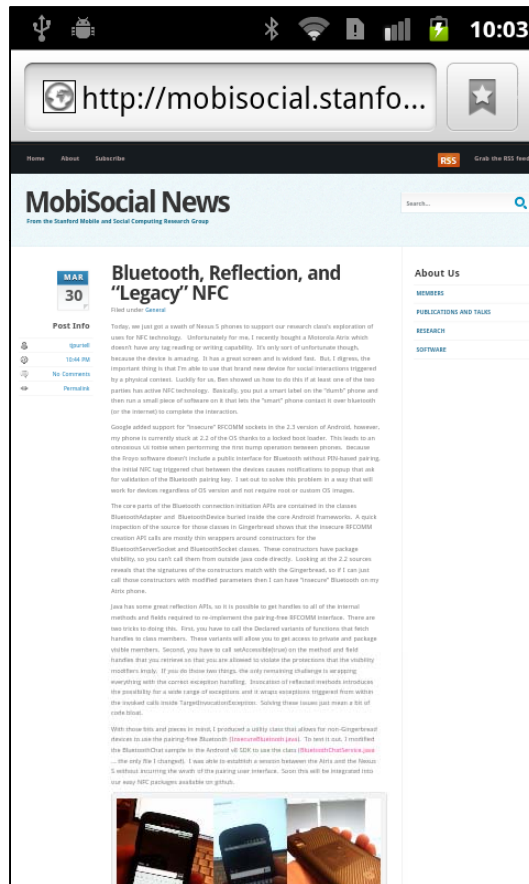
NFC is for Sharing!

Can we do better?



NFC is for Sharing!

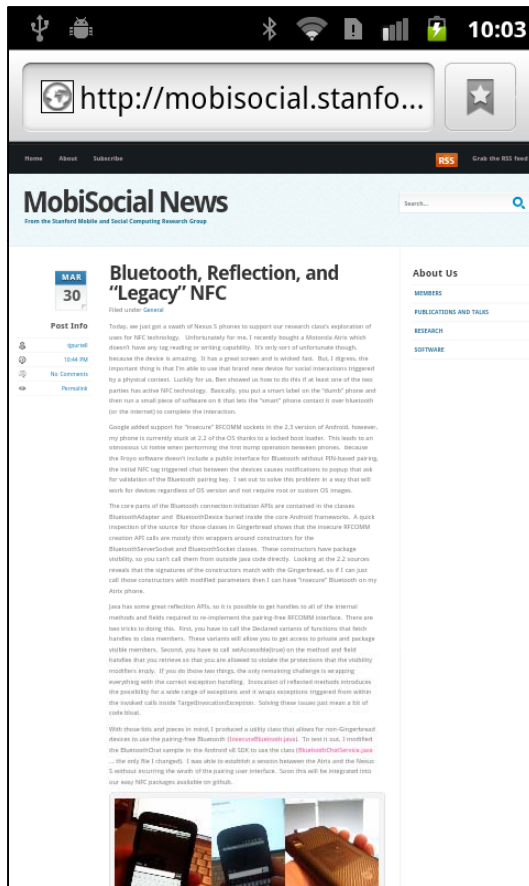
0



The screenshot shows a mobile browser interface with a status bar at the top displaying various icons and the time 10:03. The address bar shows the URL <http://mobisocial.stanfo...>. Below the address bar is a navigation bar with links for Home, About, and Subscribe, along with an RSS icon and a 'Grab the RSS feed' link. The main content area features the 'MobiSocial News' logo and a search bar. The article title is 'Bluetooth, Reflection, and "Legacy" NFC', dated March 30. The article text discusses the author's experience with a Motorola Atrix phone, their research on NFC technology, and their findings on how to bypass security measures on Android devices to use NFC. The article is categorized under 'Post Info' with links for 'Share It!', 'Like It!', 'No Comments', and 'Permalink'. On the right side, there is an 'About Us' section with links for 'MEMBERS', 'PUBLICATIONS AND TALKS', and 'SOFTWARE'. At the bottom of the article, there is a small image showing a person's hands holding a smartphone and a small NFC tag.

NFC is for Sharing!

0

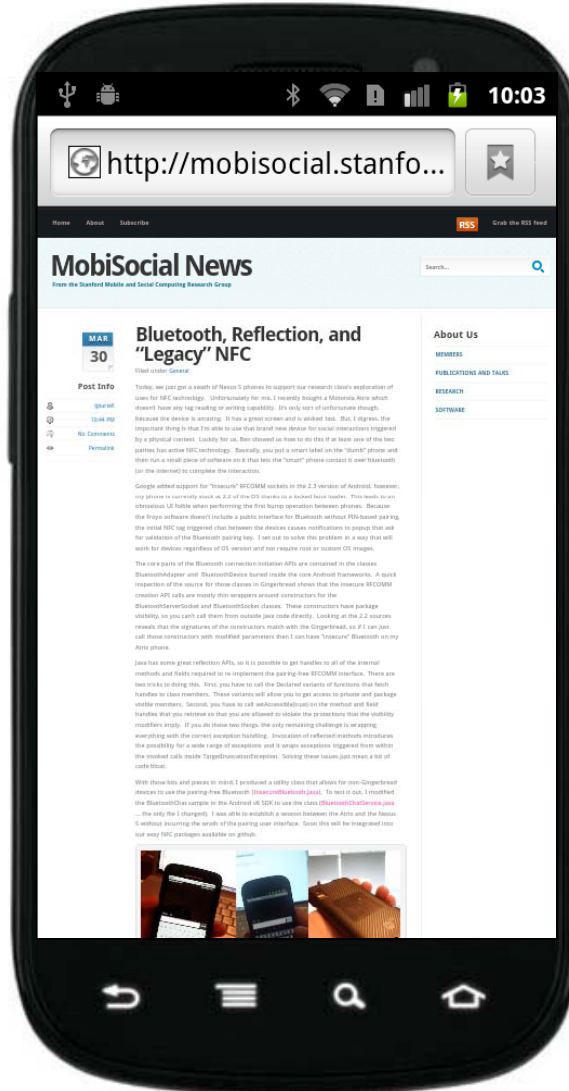


The screenshot shows a mobile browser interface with the URL `http://mobisocial.stanfo...`. The page title is "MobiSocial News" from the Stanford Mobile and Social Computing Research Group. The main article is titled "Bluetooth, Reflection, and 'Legacy' NFC" and is dated March 30. The article text discusses the challenges of using NFC on Android devices, particularly the lack of a public interface for Bluetooth pairing. It mentions that the Nexus S phone has a small piece of software on it that lets the "legacy" phone connect to use Bluetooth (or the internet) to complete the interaction. The article also discusses the use of reflection APIs in Java to access internal methods and fields, and how this can be used to bypass the lack of a public interface for Bluetooth pairing. The article concludes by mentioning that the author has created a library class that allows for non-Gingerbread devices to use the pairing flow Bluetooth (BluetoothPairingLib). To see it out, the author mentions the BluetoothChat sample in the Android v8 SDK to use the class BluetoothChatService.java.



NFC is for Sharing!

0



o-Click Sharing

Context-Aware NFC

- One touch, many interactions
 - Phone-to-Phone
 - Two people exchanging data.
 - Links, files, contact information, applications
 - Phone-to-PC
 - User-to-self interactions
 - Authenticated links, personal utilities, files, payments
 - Phone-to-TV
 - User-to-device
 - Multimedia, applications
 - Handle content with no further action