beyond device pairing: new interactions on nfc enabled mobile phones

Yaw Anokwa
Qualifying Evaluation Presentation
May 17, 2007

Hi.

I'm Yaw Anokwa and this is my quals presentation.

This is work that was done in the summer 2006 with my advisor Gaetano Borriello in conjunction with Trevor Pering and Roy Want at Intel Research in Santa Clara.

The work is called 'beyond device pairing: new interactions on nfc enabled mobile phones'

what is nfc?

- Radio Frequency IDentification (RFID)
  - Powered reader sends signal to passive tag
  - Tag powers up and responds with unique id
- Near Field Communication (NFC)
  - Range: Up to 8 inches
  - Speed: Up to 53 KB/s
  - Touch to talk

So what is NFC?

Most of you know about Radio Frequency IDentification (RFID), where you have a reader and a tag.

The powered reader sends out a signal,

Because the tag has no power, it uses the signal to power up and responds with a unique id.

Well, let's say you have an RFID reader and a tag, and you stick it on a cellphone. Maybe get another a cellphone, and make it so the two phones only talk at extremely close range.

That is Near Field Communication. NFC is a short-range magnetic induction technology like RFID, but aimed for mobile devices.

It works up to 8" and the max speed is 53 KB/s.

It's basically touch to talk. Most deployed systems about 2" of range, so it's a wireless channel that is open when devices practically touch and is closed when devices move apart.

So why is NFC interesting?

why nfc?

Technology	Characteristics
2D Barcodes	Read only, smudges and wears
Infrared	Not robust, low data rates
ZigBee	Both devices powered, low data rates
Bluetooth	Slow to pair, power hungry, expensive
Wi-Fi	Power hungry, expensive
NFC	Read/write, decent data rates, robust, low power, short range, secure, inexpensive

Well, if you just want phones to talk, there are lots of technologies to choose from.

2D Barcodes, IrDA, ZigBee, Bluetooth, and WiFi come to mind, but these have some problems.
- 2D barcodes are read only, and don't work when smudged or worn.
- IrDA requires line of sight and special light conditions. It also has slow data rates.
- ZigBee requires both devices to be powered and has very low data rates.
- Bluetooth slow to pair, both devices must be powered and it's expensive.
- WiFi is power hungry, both devices must be powered and is expensive.

What makes NFC different?
- you can read/write tags, and transfer data
- it's robust to dirt, smudges, wear/tear.
- designed for mobile, very low power and short range
- you can tie it to security controller IC
- market research says by 2011 over 30% of new phones will have nfc.

Has some downsides too so I dont want to argue. The point I want to make is that it is a very inexpensive, very lightweight, very low power technology. It fills a niche and can remove barriers to usability.

So if it's so great what deployments are available?

nfc deployments

Get information by touching smart posters!

Your NFC device is your ticket!

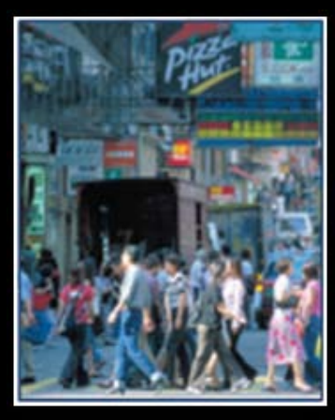
Your NFC device is your travel card!

Applications vary but are all simple and single purpose.

Can we exploit the unique capabilities of NFC and go beyond these simple single purpose applications?

Buy goods from vending machines with your phone!

Get information about your current job or task!



Your NFC device is your credit card!

The most common application of NFC is as a travel card. In fact Sony's FeliCa card is the basis of Japan's transportation card.

Future deployments that are promised are
- Smart posters
- Ticketing
- Point of sale
- Cash/credit card
- Getting bits of data

So there are a number of proposed applications. Each of these is pretty simple and serve only one purpose.

The question is can we exploit the capabilities of NFC and take the technology beyond these single purpose applications? Can we do this without breaking the usability and adoptability?

So with that problem in mind, I want to go over the outline of this talk.

outline

- Near Field Communication
  - Technology, deployments, problems
- Challenge 1: Interaction Model
  - Scenarios, related work, definitions, usage
- Challenge 2: Hardware and software
  - Description, characteristics
- Future challenges
  - Combining contributions, evaluation

I've talked a little bit about Near Field Communication. I've explained the technology, the current and future deployments, and described the problem of exploiting NFC to go beyond these single purpose applications while retaining usability and adoptability.

That problem is pretty big, so I want to identify two separate but important challenges that I worked on for quals, and show that my solutions to those challenges are valuable contributions to this problem.

The first challenge is an interaction model. I'll motivate why we need an interaction model, related work, some definitions, and illustrate where you can use it.

The second challenge is involves hardware and software to enable this model. I'll motivate why we need different hardware/software, what I've built and the characteristics of the system.

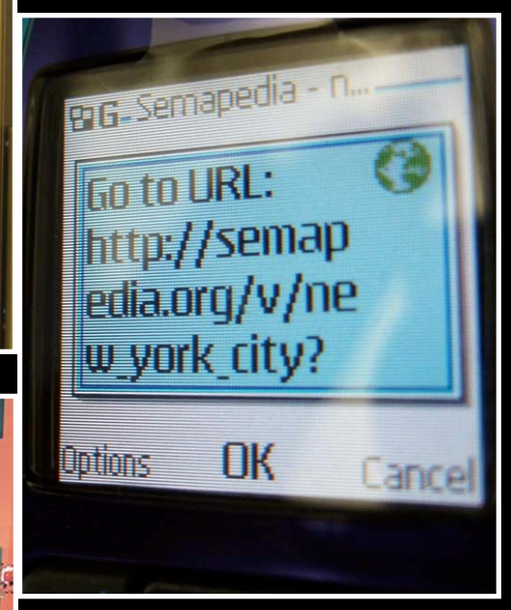
I'll conclude with how I intend on tying these two challenges together and what the future research and evaluation could be.

If you have questions at any point, please let me know.

So before we jump into the challenges, let's talk about an example that I will use throughout this talk to ground the abstract ideas.

scenario: ticketing

- See smart poster
- Scan poster with phone
- Go ticketing site (WiFi)
- Go to theater
- Watch movie



Is it really this simple? What are the problems?

NFC Forum (standards board) has a spec for a "smart poster" where you can buy a ticket. The poster is smart, because it has an embedded tag, and may some infrastructure attached to it.

So to help motivate the first challenge, let's walk through this smart poster scenario. Let's say I'm walking around on the Ave, and

I come across this really cool smart poster for The Host, a monster flick.

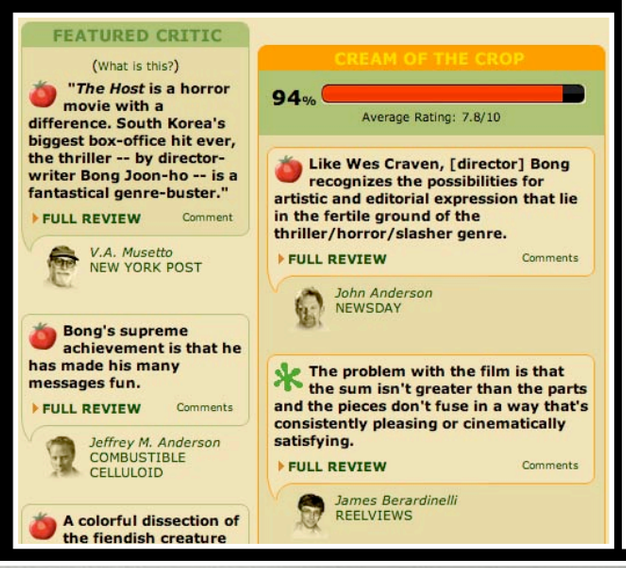
I scan the poster with my phone.
The poster gives me a URL so I go to the ticketing site and buy my ticket there using WiFi or some connection technology.
I go to the theater, show the ticket.
I then watch the movie

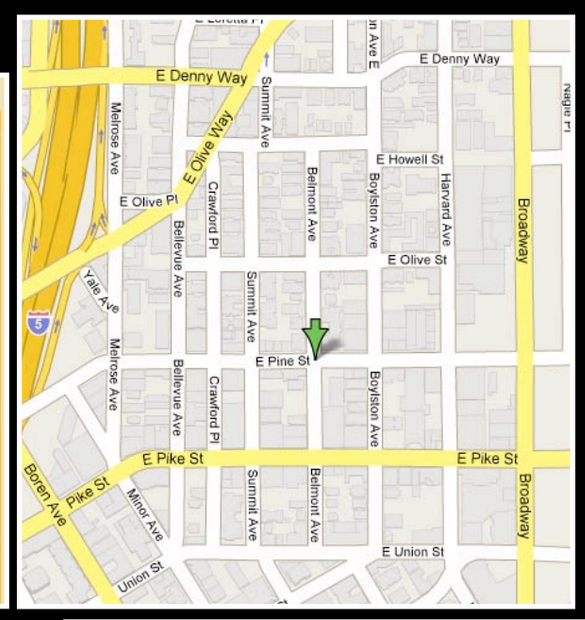
Is it really that simple? What are the problems with this scenario?

scenario: problems

Scenario ignores many of things I might want to do.

- Read reviews about film
- Get directions to theater
- Buy a ticket for a friend
- Do this without power?





Remove barriers to usability and enable these applications

Well,

Scenario ignores most things I could want to do at the poster! What if I don't want to buy a ticket? Can I still find this technology useful?

Maybe when I scanned the poster, I wanted to

- read reviews about the film
- get directions to the closest theater
- give the ticket to a friend as a gift

Can I do all this without
- without powering a poster with electricity? Can I do this on the Ave?

Again, the goal here is to remove all barrier to usability and enable these kinds of applications. By looking at previous work, we start get to ideas on how to do this.

related work

- Kostakos: NFC more engaging than Bluetooth and more efficient than 2D Barcodes. Two-way communication is required for superior usability.
- Rukzio: Users associate touch with security, speed, intuitiveness, and error resistance.
- Makela: Users make sense of the world by developing a model based on prior experience. They have vague notions about mobile RFID technology.

This isn't all the related work, but it helps motivate what a good approach is.

Kostakos and O'Neill argue that users find NFC more engaging than Bluetooth and more efficient than 2D barcodes. They identify full two-way communication in NFC as a requirement for superior usability.

Rukzio conducts a study showing users heavily associate touch with security, speed, intuitiveness and error resistance. Clearly users know the power of touching.

Makela demonstrate that most users make sense of the world by developing a mental model based on prior relevant experience. In general, they have vague notions of how to interact with mobile RFID technology. This is a key observation that we use to ground our work. Our model should help users make sense of the technology.

So then the first challenge I tackled becomes very clear.

challenge I

Interaction model for multi purpose applications

- What interface mediates between these multi purpose applications and their data?
- Does the interface help users build a better mental picture of mobile RFID technology?
- How do interactions in this model start? How do they progress? How do they end?
- How well does the model generalize across a variety of applications in this domain?

This challenge is based on the observation that if the desire is to support many applications,you have to have understand what interface mediates between these applications and the data?

Some of the other questions you may want to answer include,
- Does the interface help users build a better model of RFID technology?
- How the interactions start, progress and end.
- How well does the model generalize?

To answer these questions, we need some terminology explained.

terminology

Term	Definition	Example
Item	A physical artifact with an embedded tag	A smart movie poster
Objects	Virtual artifacts which an item represents	Reviews, directions, tickets
Actions	Operations which can be applied to an object	'Buy Ticket for Friend'

Items are a physical artifact with an embedded tag like a smart movie or poster.

Objects are virtual artifacts that an item represents or encapsulates. Examples are reviews, directions, tickets and the like.

Finally, actions are operations which can be done to object and that objects can do. An example include buy ticket for friend.

Are there any questions? Great, let's return to the scenario to explain how this model works.

scanning items

- Read reviews about film
- Get directions to theater
- Buy a ticket for a friend
- Leave notes on poster
- Poster needs no power

Manages multiple uses, removes barriers to usability

So I'm still walking around on the Ave, and I come across this really cool smart poster which beckons me. So I scan it.

When I scan this item, instead of a link to a site that you saw earlier, I get a listing of all the things I can do right there at the tag.

In this case, I can get
- information about reviews
- directions
- buy a ticket.
- I can write graffiti back into the tag for people to read.

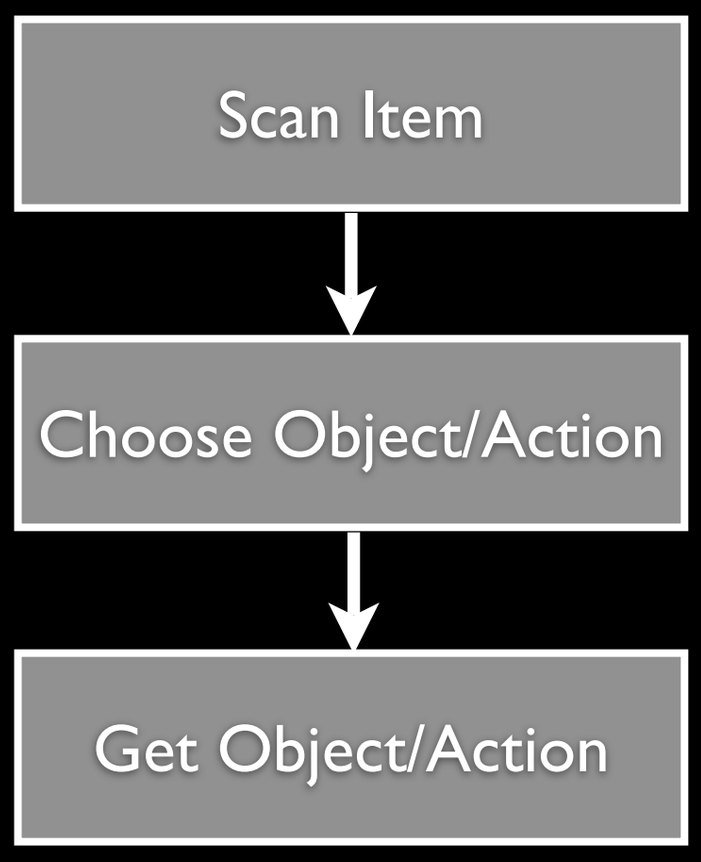
Importantly, I can do this if the poster has no power, because the phone provides the power.

This menuing method manages multiple uses for a tag and removes the barriers that users face when just touching something.

Like a good computer scientist, I can then abstract away from this example and create one part of the model: scanning an item.

scanning an item

- Scan Item: *Touch phone to poster*
- Choose Action: "*Buy Ticket*"
- Get Object: *Ticket put on phone*



How do I use an object/action on my phone?

So when we abstract away from the scenario, there are three steps I'm going through.

So when I take my cellphone and touch a movie poster, I scan an item. I get a menu of all the available stuff. So for example, the poster has reviews, directions, tickets, etc (objects and actions)

I then choose the objects or actions that I want. This can be specified by default or through a gesture or a manual selection. In this case, I have an action of "Buy Ticket" that I choose.

The get stage, moves the objects to my device. Once on device, are then stored in the appropriate folder.

Once I have the object/action, how do I use it? Let's return to the ticketing scenario.

using objects/actions

- I choose a ticket and touch my phone to Trevor's. Transfer occurs through "push".
- I touch my phone to Trevor's, swap available objects, and he chooses the ticket. Transfer occurs through "pull".

Objects can be used via "push" or "pull"

Well, I bought two tickets from that poster because I was meeting up with Trevor and I know he loves monster movies.

If you don't know Trevor, meet Trevor. He likes to have a good time.

And This is Trevor's phone. So I meet up with Trevor, and I want to give him one of my ticket. I can imagine doing this in two ways. I can first go into my phone, select the ticket, and touch his device.

The ticket is downloaded to Trevor's. In this case, I'm pushing him the data.

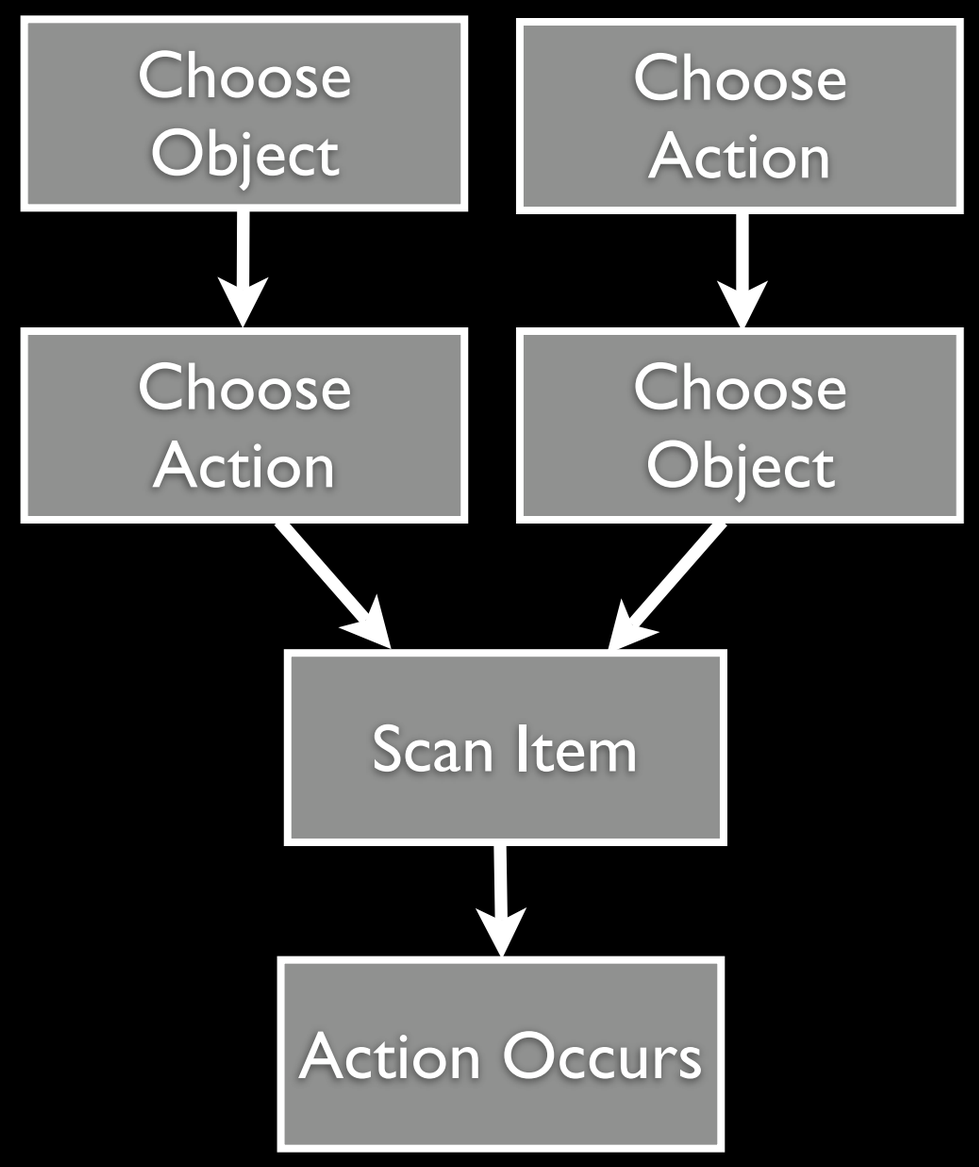
I could also just touch his device and Trevor sees all the things I have advertised, and I can see his shareable stuff. He chooses the ticket and downloads it. In this case, he's pulling the data.

So objects can be used in a push or pull.

We abstract away from this example and create the other part of the model: using objects and actions.

using objects/actions

- Choose object: *Select A ticket*
- Choose action: *Send to friend*
- Scan item: *Touch both phones*
- Action occurs: *Ticket sent*



Objects are being exchanged via "push"

So when we abstract away from, the scenario, there are two ways of looking at using objects/actions.

I can choose an object (say select a ticket) and pair it with an action (send to friend). I then touch both phones and the send to friend action occurs and the ticket is sent.

Again this is the push method.

I can also choose an action (share all) and object (all files). In this case, they are specified by default. When I scan Trevor's phone, the sharing action occurs and the ticket is sent.

Again this is the pull method.

So those that is the interaction model -- scanning an item, and using objects/actions. But a model is no good if you can't use it for other kinds of scenarios.

other scenarios

Where else do you need multiple options on touch?

- Touch camera to a printer/picture frame
- Listen to top story at newspaper stand
- Digg restaurants around the city
- Leave love notes on your girlfriend's stuff
- Grab top playlists at music store
- Kiosks, tourist directions, and more

Menus for exploration and push/pull for using objects/actions

What other scenarios can you do besides ticketing? Where else do you need multiple options on touch?

You can also imagine the interactions with the physical world.

- I can touch my camera to a printer to print my pictures or even to a digital picture frame to upload (or download) the pictures.

- Newspaper stands can advertise the top story of the day as an audio clip you can grab while running to catch the bus.

- You could have an thumbs up or thumbs down object you push to restaurants.

- You can leave NFC notes on something you know your girlfriend will scan.

- After leaving Zoka or Tower Records, you could grab a sample of the top selling songs.

- Even kiosks, tourist directions, etc are possible.

So all these can be cast in the interface that I showed to mediate between these applications and the data. The menuing could help users build a better model of RFID technology and explore tags they see. The push and pull methods for using objects/actions can be applied in these scenarios.

We can now start thinking about what we need to enable these scenarios, and that's the second challenge.

challenge 2

Software and hardware to enable scenarios

- What hardware and software support is needed to build these scenarios?
- Will the hardware and software be realistic enough for researchers to test ideas?
- How will advanced use cases (connection technologies, sensing, tweaks, etc.) supported?

The second challenge I explored was hardware and software. Essentially, what are the properties of a system that can enable Challenge 1, how do we enable the scenarios.

We also wanted to make sure the hardware and software was realistic enough for NFC researchers to test ideas.

Part of this included having support for advanced use-cases beyond what has been proposed.

Based on these questions, it was pretty clear to me that had to build our own hardware, and I want to explain why.

why more hardware?

- No NFC+Bluetooth
- No NFC+WiFi
- No NFC+sensors
- No data transfer
- Existing SDKs don't allow tweaks to the protocol
- Must be realistic for evaluation

Why build custom hardware? Aren't there phones that you can buy?

Well kind of. There was only one phone with NFC on it -- the Nokia 3220, but it was quite lacking.

The phone has no WiFi or Bluetooth and so you can't even do the common pairing applications.

Advanced use cases with accelerometer were out, and two way data transfer was also not possible.

The NFC spec provides lots of options to implement tweaks, but none of those were possible with the high level SDKs.

Finally, I could put NFC chips into a big black box, but that's not usable. Realistic hardware is important so we wanted to build something and make available to community.

host phone

- Motorola phone (E680i, ROKR E2)
- Up to 400Mhz processor
- 48MB DRAM, 80MB Flash
- Bluetooth 2.0, WiFi, SD
- C and Java ME binaries
- Linux 2.4 kernel

We use two phones to host the system

They are both Motorola Linux phones.

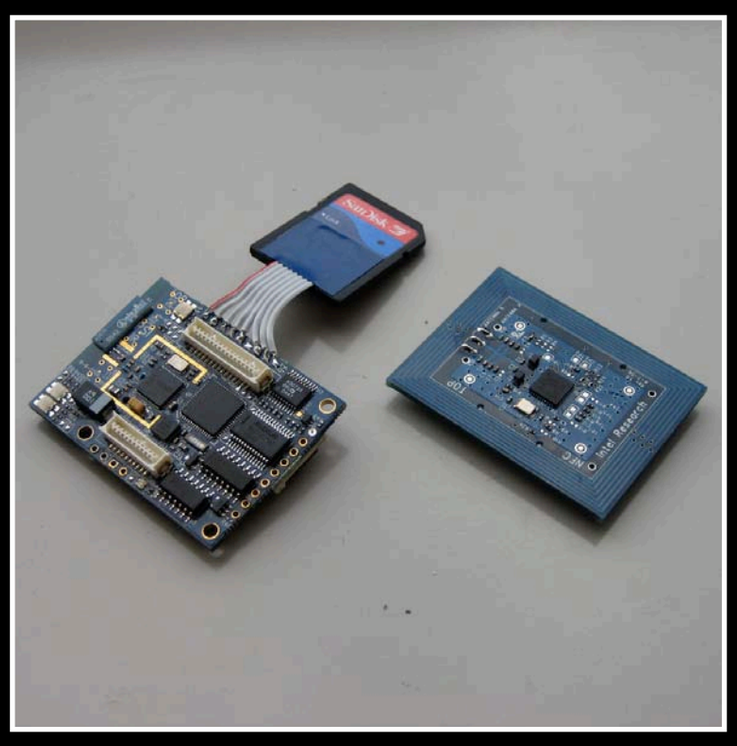
The first is the E680i, the second the ROKR E2.

Comes with 400mhz ARM
48MB DRAM, 80MB Flash
Bluetooth, Wi-Fi, and SD that supports 2Gb
We can cross compile C and deploy Java ME binaries.
Boots Linux 2.4

So to this host phone, we add our custom hardware.

psi+nfc boards

- PSI Board: controller, 802.14 radio, accelerometer, SD slot
- NFC Board: chip, antenna
- Switching between all inputs
- Hardware interfaces to SD slot via attachment
- Designed wrap around and sit on top of battery slot
- Details at BSN 2007



There are two boards that I helped build at Intel.

The PSI board and the NFC board.

The one on the left is the phone system interface (PSI). Has a micro-controller, 802.14 radio, an accelerometer, and an SD slot on back.

We use the expansion slot for the NFC board on your right. It basically has the NFC chip and the antenna.

We multiplex sensing from the SD slot, accelerometer, radio and expansion slots. It basically switches through each and feeds the phone.

The NFC board sits on top of the PSI board which feeds into a standard SD slot. The electronics aren't that complicated -- you could fit it into the same shape as a WiFi dongle (on the right).

Inside the phone, it looks like this. Board is designed to fit into battery compartment and can be covered with battery cover.

The technical details are at a paper that we published at BSN.

system tests

- Initiator: Device initiating the transaction
- Target: Device polling for transactions
- Experimental setup
  - Active mode (both devices powered)
  - Maximum speed (53 KB/s)
  - Understand data transfer

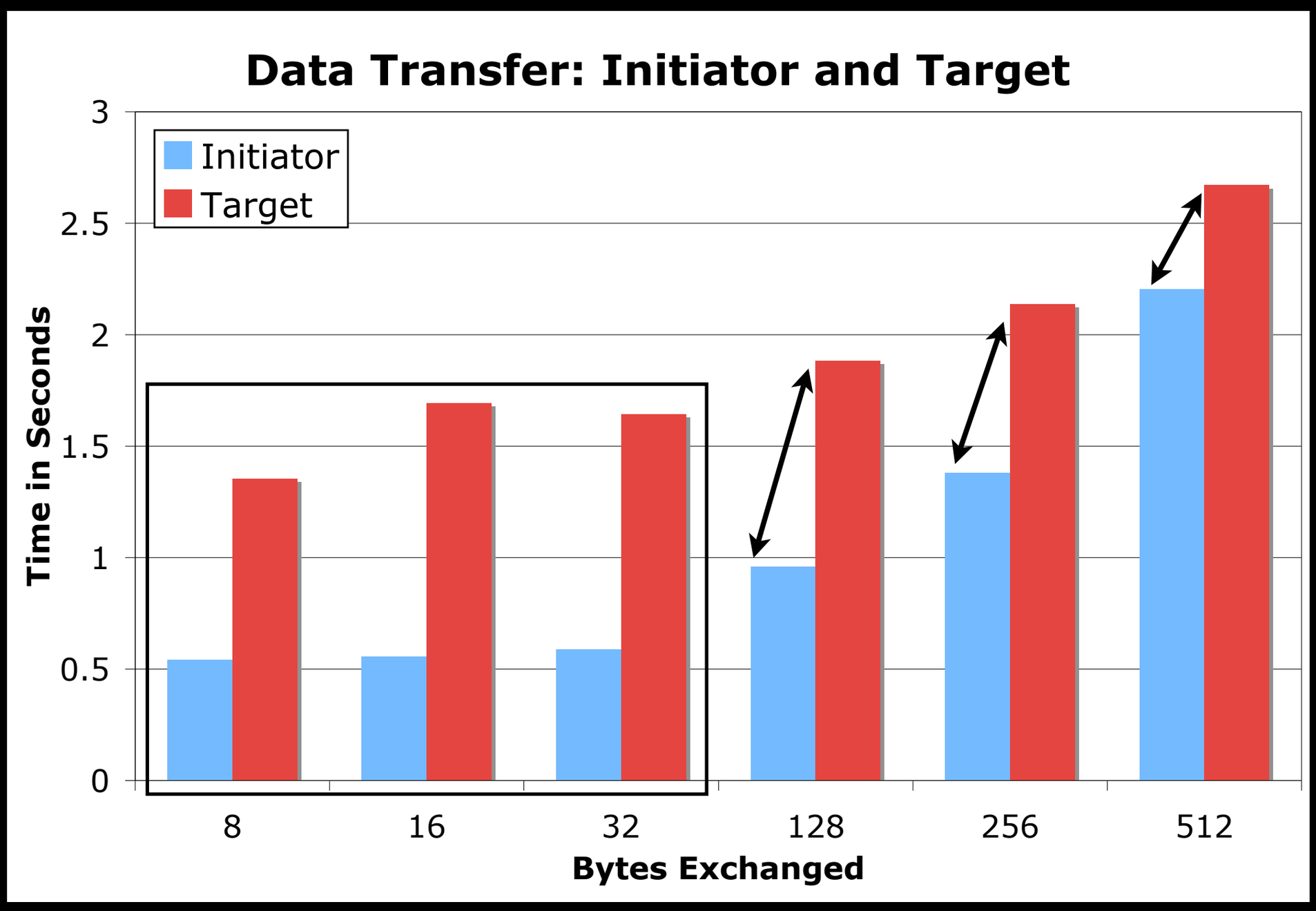
Before we get into the evaluation of the hardware, we should define a few terms.

The initiator is the device initiating the transaction.
The target is the device polling for transactions.

We run these trials in active mode at 53 KB/s and the devices are put on top of each other to prevent interference. We just want to understand data transfer.

Here is one trial run testing time to transfer and throughput.

packet size and waits



< 32 bytes, times are the same due to packet size
> 128 bytes, cost of wait time decreasing
<32 bytes gains nothing and large packets cost less

This graph will shows you where the delays between the initiator and target when two devices are transferring data. This was one trial run, but is an accurate representation.

In this graph, blue is the initiator (the device starting the transaction), red is the target (the device waiting for the transaction). The x axis has the number of bytes exchanged, and the y-axis is total time in seconds.

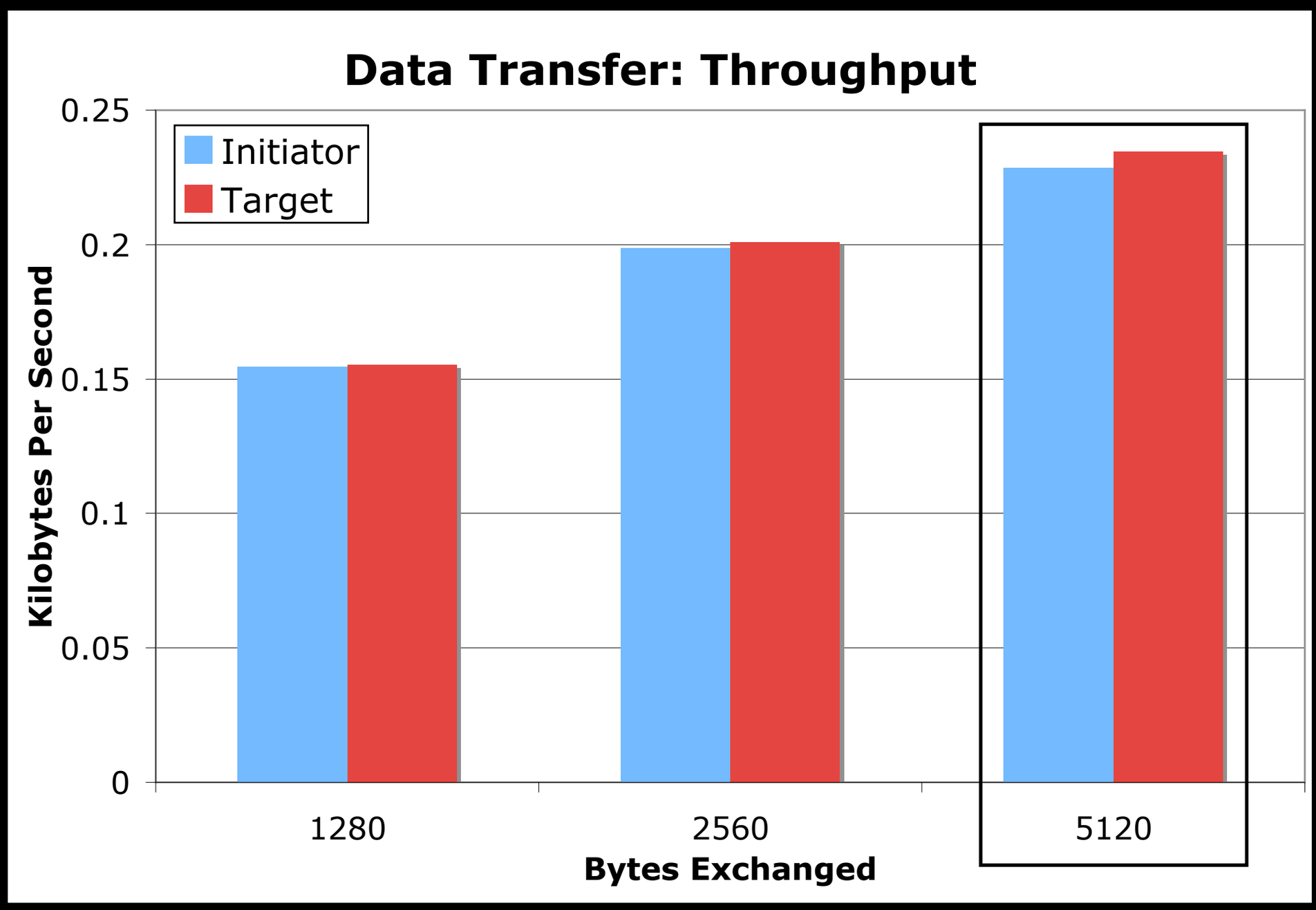
You'll notice that in general, the target (red bars) takes a bit longer than the initiator.

But if you look under 32 bytes, the times are pretty much the same. This is due to packet size. You gain nothing by sending less than 32 bytes.

As you increase amount of data exchange over 128 bytes, the packet size isn't an issue and the times start to vary. This is because the target waits to get all the data across, and with a large packet, the cost for that wait is less.

So the lesson learned in this graph is that for the best bang for buck, for a decent system staying under 32 bytes doesn't gain much and the larger the packet, the cheaper the wait cost.

actual throughput



Actual throughput is .225 KB/s not 53 KB/s

In addition to times, we also looked at throughput and this is a really interesting graph.

Again, red is the target, and blue the initiator. On the x-axis we have bytes exchanged, and on the y, we have kilobytes per second.

Throughput increases as we push more bytes in each send/receive cycle, but if you look at 5120 bytes exchanged, we still only peak at .225 kilobytes/s.

If you've been paying attention, you know the mode we set was 53 KB/s. So why are we only less than half a percent of available throughput? You would expect maybe 25%, but less than 1%?

So why the immense slow down?

why so slow?

- Switching between PSI inputs means serial stream for NFC chip must be buffered
- Implementation of firmware and driver slow data between PSI and phone
- NFC is a 'stop and wait' protocol which compounds the previous slowdowns

Slowdowns do not invalidate the work so far

The reasons for the slowdown are independent but compound.

First, if you remember the switching that we do to get all inputs from the PSI. Well, the serial stream from the NFC chip is buffered in the PSI board before being sent to the phone.

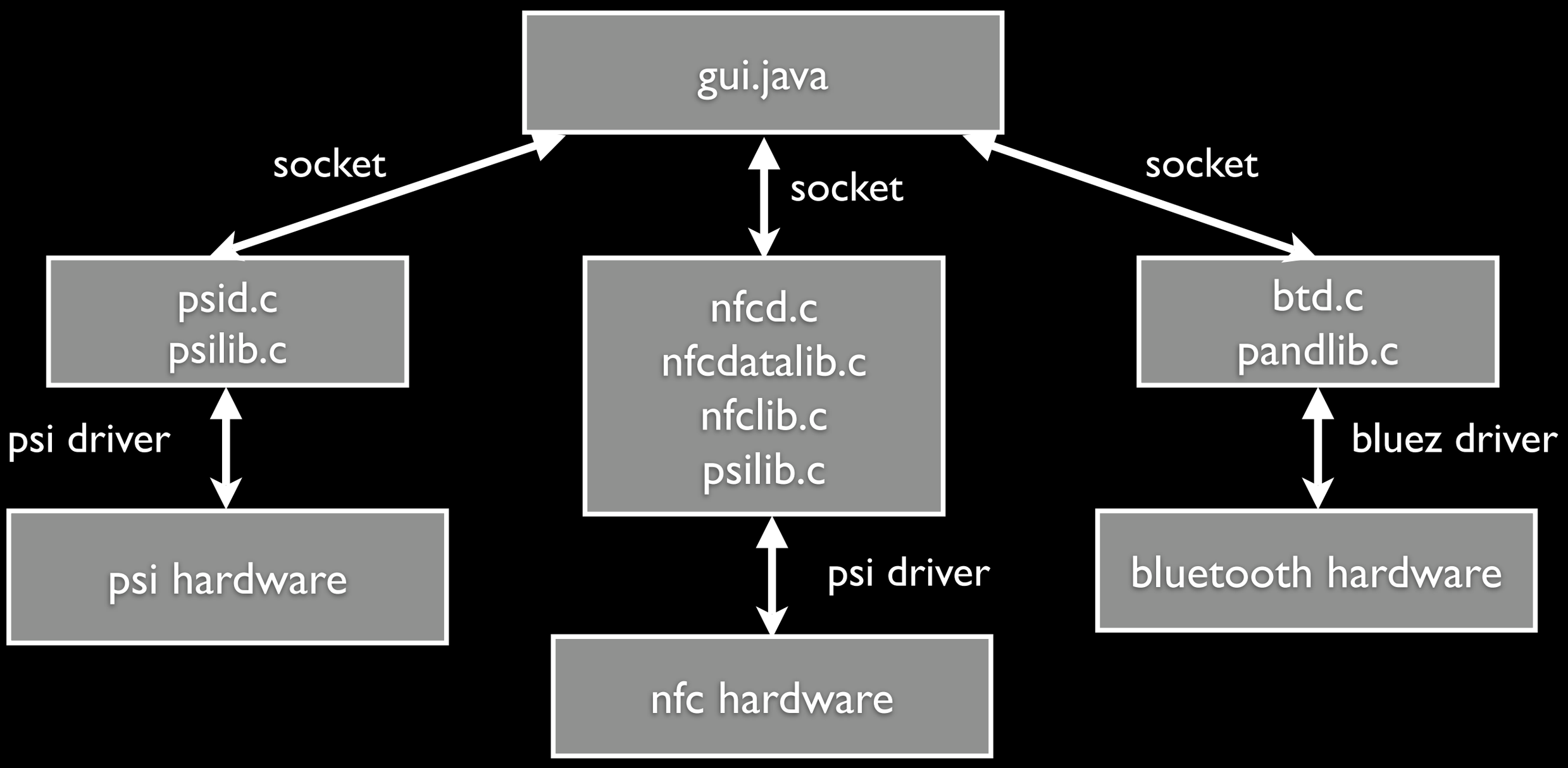
Secondly, due to how the PSI firmware and device driver were implemented, the transfer rates between the PSI and host phone itself are slow. We just don't have access too the phone's internals.

Finally, NFC is a `round trip protocol', so to handle multiple packets of data, the system must receive data, make another call to send data back, and then make the call again to receive data. This causes things to compound.

These slowdowns can be fixed, but is outside the scope of this work. We are definitely working on it and already the next generation of hardware and software is being built right now. For now, we don't feel it impacts the proof of concepts we are doing.

Now that we've talked about the hardware, I want to talk about the software a bit.

software stack



Once the hardware was in place, it took a lot of tweaks of firmware to get the phone to work. The gist of what was done was to prevent the phone from loading it's drivers and instead load the PSI kernel module.

It's easiest to look at the system in this particular configuration with three hardware devices psi board, nfc board and bluetooth devices.

The psi and nfc boards are controlled with a custom PSI driver that we developed. We also have a modified version of BlueZ (linux bt driver) to control the Bluetooth hardware.

POSIX read/write/select calls from C libraries that I wrote to talk to the device drivers. Each bit of hardware also has a daemon (that you can see at the top of each box) running to support control in both the C and Java environments.

Because the average consumer/programmer doesn't get access to these parts of the phone, I expose sockets to the Java ME midlet level so GUI apps can get access to hardware and data.

There are a couple of applications that use this software stack, so let me talk a bit about that.

our software

- GestureConnect: NFC+Sensing
  - Uses accelerometer to select objects
  - Example: Scan poster, jerk phone for ticket
  - Details at TEI 2007
- FileSharing Tool: Protocol tweaks
  - NFC scan pairs Bluetooth devices faster
  - Example: Touch phones, swap movies
  - Details at PerTec 2007

We've built a couple of apps in this framework and I'll mention two that do more interesting things with NFC.

The first bit of software we call GestureConnect which demonstrates NFC +Sensing. GestureConnect is way for you to specify some action with a gesture when you scan a tag.
- We feed accelerometer data up from the psi board to the Java ME level.
- You can then gesture (up, down, left, right) to select items in the GUI.
- You can use this software to automate the "scanning of items"

A piece of software I wrote is a file sharing tool which demonstrates tweaking the NFC protocol.
- We modify the NFC protocol to do the device pairing during initialization.
- With the tweak, we can pair devices faster.
- This software helps do the things spelled out in "using objects", but in addition to getting your transfer over NFC, large things can be done over Bluetooth as you walk away.

So with that, I want to show how I've met both challenges I discussed at the beginning and how they are contributions.

contributions

Problem: Exploit NFC while retaining usability

- Contribution 1: Interaction Model
  - Mediates multi purpose applications and data
  - Defines how interactions start and finish
- Contribution 2: Hardware and Software
  - System enables more complex applications
  - Real applications with this system possible

To recap, the big problem we were looking at was exploiting NFC while retaining usability. From there, I teased out the two challenges of an interaction model and hardware and software and motivated why each was important.

To that end, meeting those challenges are the contributions.

First, creating this interaction model. I've explain a little about how the interface mediates between these applications and the data, and showed that the menu-ing could help users build a better model of RFID technology. I walked through how interactions start, progress and finish. I also showed a few examples of how it generalizes.

The second contribution is the hardware and software. We've talked about what realistic hardware and software support is needed to build these scenarios and how will advanced use cases (connection technologies, sensing, tweaks, etc.) supported. I've also showed that applications like the ones enabled by the interaction model can be built.

Tying these two contributions is the key to future work and the evaluations I want to do.

future work

- Find more scenarios to explore where the model breaks down.
- Build more applications to discover where the software and hardware fail.
- User evaluation to test the idea and gather feedback about holes in model

I think there is a lot more work that needs to be done still

A good way to go about it is to build a couple more scenarios to see where the model breaks down.

At the same time, we need to build more applications to see if the hardware and software are enough. The software stack is being handed off to another Intel intern and they are going to push it more so that could be interesting.

Finally, one good way to tie it together is to do some user testing of these scenarios with the actual applications, and that's something we are aiming for. I think doing that will be the only way to gather feedback about the holes we have
- What happens when you have multiple devices?
- What happens when items contain many objects/actions?
- How do users to recover from mistakes?

So what those are some of the questions we are thinking about. And with that, are they any questions out in the audience?

questions?