The NewtonOS Communication Infrastructure

Endpoints, Transports and Routing

Eckhart Köppen

Introduction

- Why this talk?
- What is covered, what not?
- “Working mode of a hacker”
- Feel free to interrupt and ask!

My Motivation

- Bad handwriting – need to get data onto the Newton somehow else
- Connected world – phones, desktops, servers, ...
- Free GPRS connectivity!
- Free test phones!

Covered Topics

- Routing is quite complex
- Will cover only basics of sending and receiving
- Left out are e.g. print formats, application specific actions
- Briefly touch low level implementation topics
- Whatever comes to your mind!

Words of Caution

- The Newton is an abandoned platform – excellent documentation, but some parts are not covered
- Terminology in undocumented areas might differ
- Working mode: try yo figure out as much as needed, not more – inconsistencies, unexplained areas or errors are to be expected
- If in doubt, ask among developers

An Example

- Sending a Names entry to your mobile phone
- High level steps:
  - Pick the format for sending
  - Pick a transport method
  - Identify the receiving device
  - Open a connection
  - Send the data
  - Close the connection

Routing

- First step is choosing a data format: plain text or vCard?
- List of format choices defined by the type of the data
  - Types are referred to as “classes”
  - The class is always part of the data object
- A “Route Script” is responsible for the formatting, it claims that it can handle specific classes
- List of registered “Route Scripts” checked, those which support the current data type are listed as format choices in the Route Slip

Routing Result

- The Names entry will be formatted as a vCard, but the underlying type is text
- Other types (or classes) are e.g. views for printing or binary data
- The Route Script defines which data format (text, views or binary) it produces

Transports

- Now what to do with the chosen data format?
- The system checks how the data format can be transported
- “Transports” are responsible for taking data and sending it
- A Transport defines which data types it can handle
  - Data types relevant here are the ones produced by the Route Scripts
  - System searches list of transports for matching ones
- Transports usually implement protocols, so lets transfer the vCard over OBEX

Endpoints

- Transports need a physical connection
- Endpoints know all about that, this is the “wire” level
- Endpoint functionality is to send raw data
- OBEX in this example is interesting as it can run over different “wires”
- Here we finally cross into the low-level NewtonOS (more later)

Putting Things Away

- Receiving data is quite different
  - Common parts are Endpoints (”Wires”) and Transports (”Protocols”)
- Data will end up in the Inbox
- Route Scripts are not involved
- Instead, applications register for different data types
- The initial data type is set by the receiving Transport
  - In OBEX, the data type can be part of the protocol
  - File extensions can also be used

What Makes This So Special?

- Ideal split of the responsibilities for adding functionality
  - Easy to add e.g. new data formats
  - No dependencies between the components
  - Added functionality available for all related parts – new data formats can be used over all transports
- Nothing comparable on mainstream platform, data is boxed in
- Data on the Newton is easily accessible (Names, Dates, Notes, ...) and can be reused

The Example Refined

- How are these things implemented?
  - Route Scripts: IC/VC
  - Transports: Neo
  - Endpoint: Blunt

Routing – IC/VC

- IC/VC is a collection of route scripts and can also handle putting away items from the Inbox
- It registers to handle Names and Dates data types
- Output of the route scripts is plain text
- Every transport which can handle plain text can now send e.g. vCards
- Putting away is possible for vCard and iCal data, but the target applications are unfortunately not Names and Dates
  - Easier to implement this way
  - More a UI problem

Transport – Neo

- Neo implements parts of the OBEX protocol as a client and server
- OBEX is independent of the "wire" level
- Much code reused between the Bluetooth, IrDA and TCP/IP OBEX implementations
- Only significant differences are which Endpoint type is used – more more precisely, which Comm Tool is involved

Endpoint – Blunt

- Blunt does not directly implement an Endpoint
- Endpoints are also high level constructs
- They are layered on top of the actual physical connection components, the Comm Tools

Blunt as a Comm Tool

- The low level NewtonOS is quite different from the NewtonScript world, is is a whole set of new and undocumented concepts
- Some similarities in concepts to MacOS before X
- Blunt is realized as a CommTool, using a generic serial chip interface
- A specific serial chip is chosen on initialization, the system wide setting is done in “Bluetooth Setup”

References and Documentation

- IC/VC, Neo, Blunt: http://40hz.org/
- Blogging the hacking efforts: http://40hz.org/mottek/
- Mandatory documentation: http://www.unna.org/unna/development/documentation/
- Strongly recommended: Programming for the Newton Using Macintosh by McKeehan/Rhodes (also on UNNA!)
- Advanced but good: Wireless for the Newton by McKeehan/Rhodes