PTP/IP adapter design and connectivity techniques for legacy imaging appliances

PTP/IP Adapter Design and Connectivity Techniques for Legacy Imaging Appliances

Petronel Bigioi, Illariu Raducan, Eran Steinberg, and Peter Corcoran, *Member, IEEE*

***Abstract* — PTP/IP is an emerging connectivity standard for networked imaging appliances. It combines the ease of use and seamless connectivity of Picture Transfer Protocol (PTP) with the ubiquity of TCP/IP networks. In this paper we examine methods to enable legacy PTP appliances to gain the benefits of PTP/IP through the design of bridge and gateway adapters which can be simply plugged into the USB ports of such appliances. The internal firmware stacks of such adapters are described and practical case studies describe the adaption of legacy PTP cameras and printers to connect over a conventional 802.11 wireless LAN.**

***Keywords* — Digial Cameras, Picture Transfer Protocol, Device Connectivity and Consumer Electronics.**

I. Introduction

With the wide adoption of new wireless technologies it was clear that sooner or latter the digital still camera will become a networking device. A new transport protocol, PTP/IP was adopted as a CIPA standard in 2005 [3]. The new transport is described in detail in [4] and it enables PTP [1], which could previously only be used with USB device connections [2], to be used with TCP/IP networks. The PTP/IP standard allows extension to a network environment without affecting applications that use PTP. It also enables multiple simultaneous connections among digital imaging devices, which was not possible with PTP over USB.

With PTP [5-7], digital cameras can exchange images with host computers, printers, other imaging and display devices. The interoperability resulting from the widespread adoption of PTP over USB transport has greatly reduces consumers' frustration at the difficulty of transferring photos from camera to computer, printer, or kiosk, and contributed to the ongoing growth in sales of digital cameras and associated devices.

While PTP/IP enables next generation products to function over 802.11 home WLANs there remains a large installed base of legacy PTP appliances. With the advent of PTP/IP the users of such cameras and printers may feel unhappy that these recently purchased products are already obsolete and may be reluctant to pay the cost of upgrading to a new PTP/IP compatible printer or camera. In turn this may slow the adoption of PTP/IP in the market.

Bearing these factors in mind a decision was taken to investigate the feasibility of creating low-cost USB adapters with 802.11 connectivity which could convert legacy cameras and printers into PTP/IP compatible appliances. This would allow existing users of PTP products a low-cost upgrade to convert these legacy appliances and gain the benefits of full device connectivity over a home WLAN. As generic low cost USB to WLAN adapters are available the key design problems relate to the additional adaption which is required to the internal firmware stacks of these adapters in order to support the additional signaling requirements of PTP/IP – in particular the requirements for *device discovery* and *device bonding* which are inherent in PTP/IP.

· Petronel Bigioi is VP of Engineering at FotoNation (Ireland) Ltd.; e-mail: petronel@fotonation.com; Illariu Raducan is a Senior Engineer at FotoNation (Ireland) Ltd.; e-mail: lale@fotonation.com.

Peter Corcoran is with the Faculty of Engineering at National Univeristy of Ireland, Galway; e-mail: peter.corcoran@nuigalway.ie.

Eran Steinberg is CEO of FotoNation Inc.; e-mail: erans@fotonation.com.

II. System Design

A number of alternative WLAN chipsets and SoC hardware platforms were investigated. It was found that hardware platforms where there is support for open source software tend to be easier to work with and key adaptations of the firmware to the underlying hardware tend to more easily achieved. It is also worthwhile mentioning that open-source TCP/IP software stacks tend to be quite robust and somewhat easier to customize.

Initial prototyping was achieved using a commodity wireless access point to which a full open-source operating system had been ported. This provided complete hardware level access to a mass market chipset for negligible set-up costs and was a key factor in completing initial product feasibility and reliability studies.

Some firmware modifications were required in the adapter firmware over and above the normal adaption of a USB communications stack to provide TCP/IP network connectivity. These included the addition of (i) a *device discovery* mechanism and (ii) a *device bonding* mechanism. We will next consider a number of options which were employed for these, describing the pros and cons of each.

A. Device Discovery Mechanisms

An important component of all USB based networking protocols, including PTP, is their *Plug and Play* functionality. This is also a requirement of PTP/IP although its implementation is not discussed in the PTP/IP specifications. The TCP/IP protocol does not inherently support the necessary services for implementing the equivalent of a interrupt service to signal that a new device has joined the network, thus it was necessary to consider some of the extensions that have been employed by other TCP/IP based protocols and to investigate practical implementations.

1) Manual Configuration

The simplest way of dealing with device discovery is to manually configure the Initiator with the IP address and

port number of the Responder device. The Initiator will then perform various types of tests to figure out when the configured Responders will appear on the network by periodically trying to establish PTP-IP connections or using network specific checks, such as ping.

This method will only work in statically configured networks where devices have predefined IP addresses and would be unworkable in networks where Dynamic Host Configuration Protocol (DHCP) or AutoIP are employed for IP address management. As most networks, particularly home networks, are likely to use DHCP this approach is not very useful for CE applications.

2) DNS Service Discovery

DNS Service Discovery (DNS-SD) is Apple's discovery protocol and it is considered lighter and simpler than other competing discovery protocols because it uses DNS as mechanism to transport the data, rather than other more complex protocols such as HTTP.

DNS-SD may be used to find services, such as digital cameras, printers, without a directory server. It is usually implemented in conjunction with multicast DNS (mDNS) which is used to translate between names and IP addresses without a DNS server. Thus it employs the DNS SRV, TXT and PTR [10] records to advertise services (service type, domain name and other configuration parameters). Note that DNS-SD is not dependent on mDNS, but these are complimentary technologies that function well when used together. Thus our implementation was initially tested in an mDNS environment.

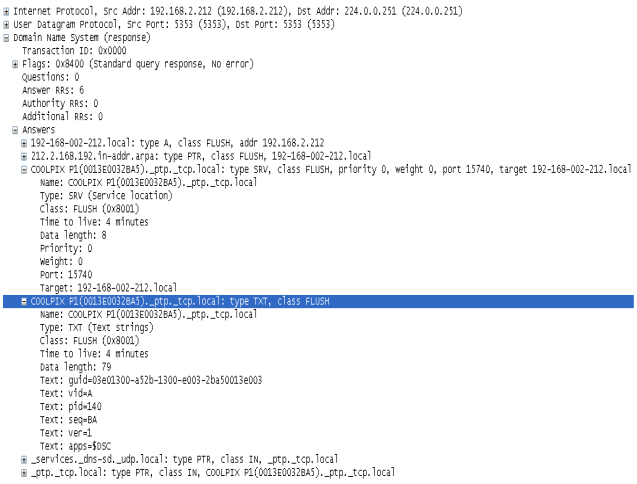

Fig 1: *Example DNS-SD response from PTP/IP compatible camera.*

While it is an informative rather than normative section, the PTP-IP specification details the required and optional fields to be used by a PTP-IP Responder using DNS-SD as the service discovery protocol. The PTP-IP Responder should register its presence with the SRV record "_name._ptp._tcp.local.", IP address and port number.

Of the TXT records that may be registered, the most useful is the *global unique identifier,* or GUID of the device. The PTP-IP Initiator will enumerate all devices in _ptp._tcp.local. sub-domain and can use associated TXT records for additional filtering and selective notification generation. ***Fig 1*** presents an example for a real digital camera using mDNS and DNS-SD for device discovery, as advised in [11].

3) Universal Plug and Play Discovery

Universal Plug and Play (UPnP), is a set of networking protocols promoted by the UPnP forum [12] that allows devices to interconnect seamlessly and allow for easy networking configuration, especially in a home network environment. UPnP provides standards for automatic networking configuration, automatic device discovery, service description, control, event notification and service presentation. It is integrated into the operating system of most desktop computers and as such provides a significant installed base of compatible networked computers. Thus basing a discovery protocol for PTP/IP on UPnP is a very pragmatic approach.

The discovery portion of UPnP is based on the Simple Service Discovery Protocol (SSDP) [13]. SSDP provides a mechanism that network clients can use to discover network services with little or no static configuration. SSDP provides multicast discovery support, server-based notification, and discovery routing. Typically, an imaging device should implement at least the basic UPnP device profile, in order to be "discoverable" by desktop computers or other UPnP compatible appliances. This is also necessary to enable application programmers to employ standard operating system APIs in order to retrieve network arrival/leave notifications.

When a device appears into a network, SSDP allows that device to advertise its services to control points on the network. Similarly, when a control point is added to the network, SSDP allows that control point to search for devices of interest on the network. The exchange is done using a discovery message formatted using HTTP known as an "alive" packet. This contains a few essential specifics about the device or one of its services, e.g., its type, identifier, the time for which the service is active, and a URL to access for more detailed information. The device announces its intention to leave the network by using another type of SSDP message, called a "byebye" packet. More detailed descriptions can be found in ref [14] and practical examples of camera responses are shown in ***Figs 2(a) & (b)*** below.

The most important information contained in the "Alive" message is the URL that points to the location of an xml document that contains the description of the device. This device description xml document can be retrieved by all interested parties via the HTTP protocol. A disadvantage is that the device is required to implement a mini-HTTP server, although this is not an excessive requirement for today's embedded devices.

The device description document can have a list of services, each of them having an associated URL, and interested parties can retrieve the service description xml document, containing detailed information about specific services offered by the device. UPnP offers significant flexibility in its service discovery, but at the cost of greater complexity. In turn this may lead to less reliability. Our

initial feasibility testing suggests that SSDP (and the additional UPnP components that will be required to properly implement it) would not offer the most appropriate solution to meet the requirements of the majority of device manufacturers and consumers. However, given that Microsoft is promoting the MTP/IP protocol, essentially a superset of PTP/IP, we feel that SSDP support for PTP/IP is likely to be mandatory for many applications.

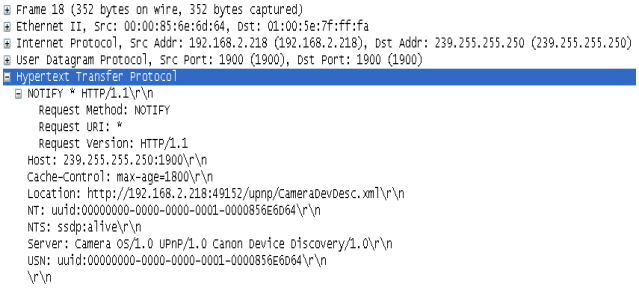

Fig 2(a): *Example SSDP "alive" response from PTP/IP camera.*

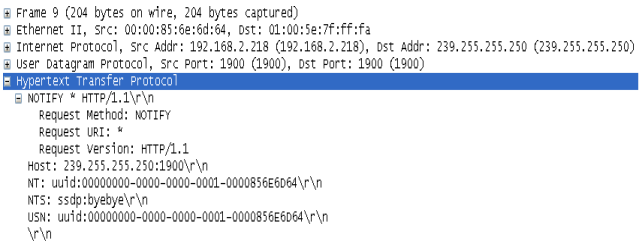

Fig 2(b): *Example SSDP "byebye" response from PTP/IP camera.*

4) A Customized PTP/IP Discovery Protocol

Following on from our preliminary tests we felt that the available candidates for service discovery protocols did not completely meet the requirements that device manufacturers or consumers might wish for. In particular we felt that there was scope for a specialized, lightweight implementation which did not rely on high level networking services such as DNS or UPnP. Such a lightweight protocol serves a similar role however its targeted scope is restricted to the discovery of other imaging devices that support PTP-IP.

It can be employed in the place of standard protocols like UPnP and Rendezvous and is useful where the complexity of these standardized service discovery approaches is beyond the capabilities of an imaging appliance, or indeed where the manufacturer chooses to implement a less complex service discovery mechanism in the interests of device reliability. The protocol employs a UDP broadcast mechanisms and can be used in conjunction with TCP/IP stacks that do not implement multicasting. A more detailed description is provided in section III below.

B.Device Bonding Mechanisms

As many imaging devices will not typically stay active on a network, but will periodically leave and rejoin the network there is a requirement for a device bonding mechanism in addition to a device/service discovery mechanism.

The simplest usage case is a network with two image printers – one being a high volume commodity printer, the second a low volume high quality printer. It would be typically desirable to bond a consumer grade digital camera or a camera-phone to the commodity printer and a professional or prosumer grade camera to the high quality printer. Thus a user could ensure that only high quality images would be printed on the high quality printer without needing to manually check the network connections and device associations every time.

There are three principle approaches to implementing device bonding: (i) pre-configured bonding, (ii) dynamic bonding and (iii) a custom PTP/IP pairing protocol.

A fourth approach is *open bonding* where all responders can bond to an initiator. Clearly this requires that an initiator has the ability to support multiple simultaneous responders and it is up to individual responders to determine whether they wish to bond or not. This is, in effect, a broadcast model and is somewhat restrictive in most practical use cases.

1)Pre-configured Bonding

Devices using this method implement a default rejection policy for devices that are not know. The known device list is built offline using a custom approach. PTP-IP protocol allows for the transportation of device GUID (from Initiator to Responder and the other direction too), so both the Responder and Initiator can take decisions to allow or reject connection requests.

One example of such approach would be to use a USB connection between an Initiator and Responder (i.e. PC to Camera) in order to configure the list of the allowed GUIDs in the Responder and to read the Responder's GUID that should be detected and reported to the application level. Another approach would be to store/read some special information using a card that would be moved between the Responder and Initiator. Although there are some security advantages to this approach it is more burdensome to users than the alternatives and the least attractive approach for consumer applications.

2) Dynamic Bonding

This approach can be implemented on devices with more sophisticated user interfaces and it is built on the feature of PTP-IP protocol where the exchange of *global unique identifiers* (GUIDs) between devices occurs during the PTP/IP connection establishment phase. The implementation of PTP/IP in the *responder* transport can pass the GUID information to the application level, allowing the device to request user confirmation to enable a connection or to check a *permitted device* list. Similarly, on the *initiator*, a network camera setup application can allow the user to browse the network for *responder* GUIDs and select one. Typically device GUIDs will be linked with *friendly names* for devices, so a user would browse through a list of named printers or other image related network services (e.g. image archival, classification or sharing services are typical examples).

3)PTP-IP Custom Pairing Protocol

This protocol has been developed specifically for PTP-

IP devices that have no user interface. The only requirement is for those devices to have a way to enter a special *pairing state* for a short period of time (e.g. a simple button). The main idea is for the user to perform asynchronous action (e.g. button press) on both devices that are to establish a PTP-IP connection. At the end of the *pairing interval*, both devices should have detected exactly one peer device and a peer-to-peer bond is established.

III. PTP-IP Custom Discovery Protocol

The specialized PTP-IP discovery protocol serves the role similar to what other service and device discovery protocols do: it helps to find devices and services in the networking environments. Its targeted scope, however, is limited to discovering imaging devices that support PTP-IP.

This discovery protocol can be used instead of standard protocols like UPnP and Rendezvous when the complexity of the standard approaches is beyond the capabilities of the devices or when other protocols cannot be used for various reasons. The protocol is based on UDP (User Datagram Protocol) broadcast mechanisms and can be used in conjunction with TCP/IP stacks that do not implement multicasting. Although UDP broadcast may put an unnecessary burden on a large network, this method may work quite well in the scenarios covering the home network usage of digital cameras and other imaging devices.

The specialized PTPIP discovery protocol uses two types of messages: Device ANNOUNCE messages and Service DISCOVERY messages.

1) Device ANNOUNCE messages

Are broadcasted onto the local network by a device that has arrived into the network (i.e. a wireless digital camera).

There could be two types of such messages: the arrival of a new service of a certain type and the termination of a service.

2) Service DISCOVERY messages:

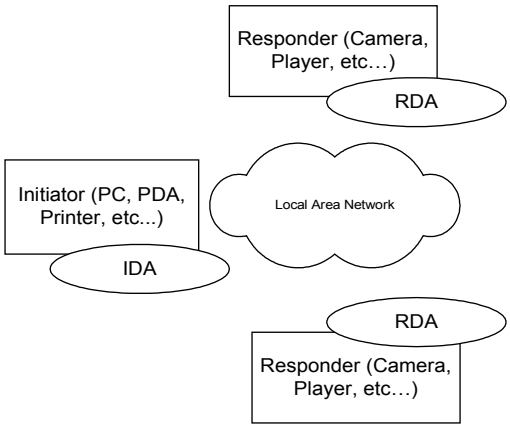

Fig 3: PTP-IP Custom Discovery Protocol Model

These are broadcast onto the local area in order to discover services of a certain type existing in the network (e.g. a PC discovering all the PTP wireless digital cameras in the network).

The receiving device will respond with a unicast announcement message, identifying the protocol it implements, IP address and port number where this service is available. Optionally, the devices could advertise a friendly name that can be used in a user interface.

The PTP-IP Initiator will need to implement the Initiator Discovery Agent (IDA, Figure 4) – the software that will perform the following tasks:

- Start – will send out DISCOVERY messages to build up a list of available PTP services
- Normal Operation – listen for ANNOUNCE messages and maintain a list of connected devices
- Stop – No special packets need to be sent or received upon the exit of this software.

The PTP-IP Responder will need to implement the Responder Discovery Agent (RDA) as illustrated in ***Fig 3*** - the software that will need to perform the following tasks:

- Start – send an ANNOUNCE ARRIVAL message. Each ANNOUNCE ARRIVAL message will have a field specifying a time interval. This time interval represents the time after which the digital camera will issue another ANNOUNCE ARRIVAL message. This will be used in the implementation of the IDA to detect a device that failed to re-announce its presence due to some internal or networking problems.
- Normal Operation – listen for DISCOVERY messages and respond to those via a unicast with an ANNOUNCE ARRIVAL messages. Send (broadcast) ANNOUNCE ARRIVAL messages before the time interval specified by the previous message has elapsed.
- Stop – send an ANNOUNCE LEAVE message to let any listening hosts know that the PTP-IP service is about to become unavailable.

IV. Custom PTP/IP Device Pairing Protocol

This protocol has been developed specifically for PTP/IP devices and is particularly useful in the context of pairing two such devices that have little or no user interface. The only requirement is for those devices to have a way to enter a special *pairing state* for a short period of time. This can be achieved using a simple button-press. A simple indication that the pairing process has completed successfully is also useful and could be realized using a simple LED or audio tone.

The protocol is based on exchange of UDP broadcast messages, that contain the GUID and friendly names of the devices which are to be paired. The main idea behind this protocol is that that the user will perform a relatively synchronous action on both devices, at least in user-time. Thus each device will remain in its *pairing state* for a sufficient time to allow the user to activate both devices. Typically we suggest 10-20 seconds is a suitable timeframe – in a worst case scenario a user should be able to activate the pairing switch on both devices within 4-5 seconds.

A.Pairing Procedure

An exemplary sequence of device messages is illustrated in ***Fig 4*** below. The first device to be actuated is D2 which broadcasts an INFO message on UDP port 15740. This contains the GUID and *friendly name* and may optionally contain manufacturer specific data. Note that the first such message is likely to arrive at D1 before that device is set into pairing mode, but this message will be rebroadcast at regular (but slightly randomized) intervals until a confirmation message is received from D1.

When D1 is actuated its first broadcast message collides with the second broadcast message from D2 and both are lost. However the second broadcast from D1 is successfully received by D2 which issues a CONFIRM message. Similarly the third INFO broadcast from D2 is successfully received and confirmed by D1 completing the pairing process.

Note that confirmation messages are also rebroadcast for a fixed number of regular (but slightly randomized) intervals. We recommend at least 3 such confirmation rebroadcasts. Typical rebroadcast intervals are of the order of 100 ms with a randomization interval of +/- 20 ms.

At the end of the pairing timeout interval, both devices involved into the paring process should have received CONFIRM messages from exactly one peer device. If this condition is not satisfied, than the pairing protocol will fail and some indication should be given to the user who may opt to retry the process.

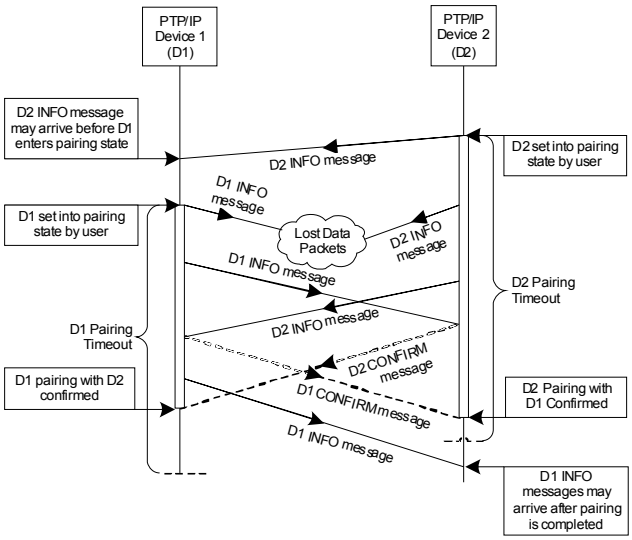

Fig 4: *Custom PTP/IP Pairing Sequence*

B. General Message Format

Each packet part of this protocol MUST start with a 16 bytes field that would represent a globally unique identifier of the protocol – a unique value specially generated for this protocol. The GUID we recommend for use for implementations of our PTP/IP pairing mechanism is 0FDB-8EFB-6968-4734-A0CF-4869-4382-A3D7.

This GUID is basically a protocol identifier and guarantees that the received UDP packets will be correctly interpreted by other pairing-enabled devices. The general message format is presented in Figure 3.

Field	Size [Bytes]	Data Type
Protocol GUID	16	UINT8
Packet Type	2	UINT16
Payload	Unknown	Unknown

Fig 5(a): *General Packet Format*

Note that the packet type may have a value of 0x01 for INFO messages and 0x02 for CONFIRM messages. The payload must be considered of variable size to allow for the inclusion of manufacturer specific data.

C. DEVICE INFO Message

This message MUST be sent by both devices wanting to complete the pairing. The message will be broadcasted on UDP port 15740 upon entering the pairing procedure. During the pairing mode, the PTP/IP device should send one DEVICE INFO Message every *time_period* interval. The *time_period* interval should be of the order of one second. The expiry time of pairing procedure must be set to at least 5x the *time_period* and preferably 10x or greater.

Field	Size [Bytes]	Data Type
Protocol GUID	16	UINT8
Packet Type	2	UINT16
Device GUID	16	UINT8
Device Friendly Name	2 ... 80	UINT16
Reserved Data	0 or more	UINT8

Fig 5(b): *Device INFO Message Packet Format*

Note that the Device Friendly Name is a *null-terminated* UNICODE string that contains a human readable name for the remote device. This field and should be used only for UI purposes. The Reserved Data field comprises of 0 or more bytes reserved for the expansion of the protocol. The maximum value is limited by the MTU (Maximum Transmission Unit) of the used underlying network. Typically this will contain manufacturer specific custom data.

D. CONFIRM Message

In response to a device INFO message, a PTP/IP device should issue at least a unicast message on UDP port 15740, having the format presented in ***Fig 5(c)***. It is recommended that the device send a sequence of at least 3 CONFIRM messages, at a time interval of 100ms.

Field	Size [Bytes]	Data Type
Protocol GUID	16	UINT8
Packet Type	2	UINT16
Own Device GUID	16	UINT8
Received Device GUID	16	UINT8

Fig 5(c): *Confirm Message Packet Format*

V. Practical Adapter Embodiments

So far we have considered the additional protocol components required in order to realize a working PTP/IP adapter for legacy devices. In this section we present two of the simplest case studies for (i) a PTP/USB digital camera and (ii) a PTP/USB printer conforming to the CIPA-DC-001-2003 PTP printing standard. It is worthwhile remarking that the combined

initiator/responder stack evident in both of these case studies suggests a new form of peer-to-peer device architecture. However this lies beyond the scope of the present work and will be presented elsewhere.

A. PTP Camera Adapter

This can be either an internal or an external PTP/USB to PTP/IP device that network enables an existing USB digital camera which supports the PTP protocol. When implemented externally, the adapter takes the form of a dongle which plugs into the USB port of the camera. The camera only runs PTP protocol over USB, while the PTP adapter acting as either a bridge or a gateway runs all the required communication protocols to make the camera behave correctly as a PTP-IP network camera. One end of the adapter acts as a USB-Initiator and is connected to the camera, and another end is a PTP/IP-Responder and is connected to a PTP/IP Initiator.

Where the camera implements a DC-001 compatible [8] client, then an adapter which can support a single PTP session is adequate and a simple PTP Bridge is implemented in the adapter firmware. The protocol for selecting the PTP device with which the camera communicates during this session is implemented within a device discovery layer which is responsive to a user actuating a pairing actuator (not shown) on the adapter and an associated printer within the required time out period and without interference from other pairing devices.

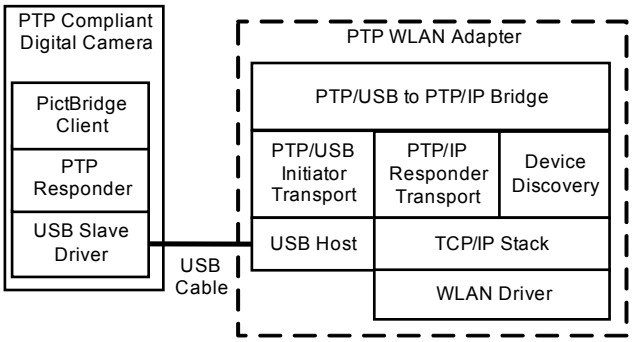

Fig 6: *Protocol Stacks for PTP Camera Adapter*

B. PTP Printer Adapter

An internal or external PTP/IP to PTP/USB device that network enables an existing USB DC-001 compatible [8] Printer. Again, when implemented externally, the adapter takes the form of a USB dongle. The printer runs DC-001 compatible [8] over USB, while the PTP adapter running as a gateway runs all the required communication protocols to make the printer look like a network DC-001 compatible [8] printer. In this case one, end of the adapter is a USB-Responder and is connected to the printer, and another end is a PTP/IP-Initiator and is connected for example across a WLAN to a WiFi camera.

Such a translator allows a currently available standard DC-001 compatible [8] printer to talk with PTP/IP cameras that have recently become available on the market.

This printer adapter has two interfaces: a WIFI interface and a USB Slave interface. Its primary function is to transform a standard USB DC-001 compatible [8] printer into a WIFI PTP/IP compatible printer, available on a wireless local area network. Such an adapter preferably connects only to PTP/IP compatible clients (i.e. PTP/IP Responders that advertise their application protocol to be compatible with DC-001 [8], in other words, their intent to print).

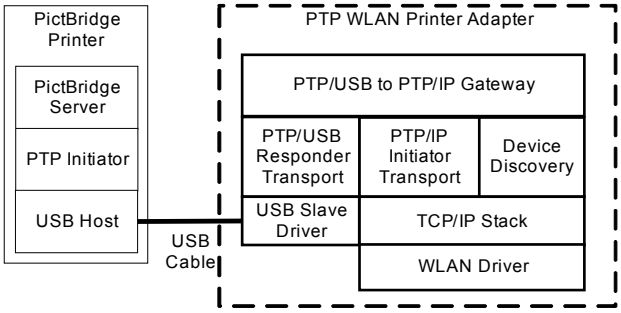

Fig 7: *Protocol Stacks for PTP Printer Adapter*

In practice the adapter can only be implemented with a gateway layer rather than with more basic bridge functionality. One reason is that the legacy printer side USB transport (unlike PTP/IP) does not carry enough self-descriptive information about the command being transported, so it does not allow the PTP Printer Adapter to be implemented as a simple PTP Bridge.

Since only one remote device connected to the PTP printer adapter can print to the actual printer at a time, the printer adapter deals with multiple paired PTP/IP devices in sequence beginning with the first paired device for which print intent has been detected. Once the print is done with this device, the dongle looks for more paired PTP/IP devices with print requests. If more devices are detected, the printer chooses one and initiates the connection to them.

Due to limitations of the PTP/USB transport specification the translation that takes place between PTP/USB and PTP/IP can't be done just at the transport level. The device will need to be a PTP Gateway, in the sense that it must interpret the PTP payload to figure out what command is in progress. In this way, the translator will be able to associate a command with a known data-phase by maintaining a lookup table with the required associations. Note that such a device will not work with vendor specific commands,unless the translation table is updated with vendor specific command information, for each vendor command that is to be supported.

VI. Conclusions

In this paper we have reviewed the two principle aspects of the PTP/IP standard which are left open to vendor-specific implementations, the device discovery process and the device bonding mechanism.

We reviewed a number of industry standard discovery protocols discussing the pros and cons of each and have also presented a custom PTP/IP discovery protocol which can be employed where device resources are constrained

by cost or design issues. Similarly we discussed a number of approaches to device bonding including a custom pairing protocol which is optimized for PTP/IP applications.

Finally we presented a protocol stack structure which enables legacy PTP devices to behave as networked PTP/IP devices through a matched initiator/responder protocol stack. For one-to-one mappings a simple bridging layer can be used to glue these matched stacks together. For more sophisticated devices such as printers which may support multiple devices connections and more sophisticated command sets a "smart" gateway layer is required to perform command interpretations and to manage multiple device connections.

This matched protocol stack concept is quite powerful, enabling device vendors to quickly modify their legacy PTP products to behave as network aware PTP/IP appliances. The concept could be further developed to provide a proxy-style services gateway for networked PTP/IP devices.

References

[1] ISO 15740, "Picture transfer protocol (PTP) for digital still photography devices", *Photography - Electronic still picture imaging*, 2005

[2] USB Implementer's Forum, "Universal Serial Bus Still Image Capture Device Definition Revision 1.0", July 2000, http://www.usb.org/developers/devclass_docs/usb_still_img10.pdf.

[3] Camera & Imaging Products Association (CIPA), "PTP-IP Picture Transfer Protocol over IP version 1.0", Japan, 2005. http://www.cipa.jp/hyoujunka/kikaku/pdf/DC-X005.pdf,

[4] P. Bigioi, G. Susanu, E. Steinberg, and P. Corcoran, "PTP/IP - a new transport specification for wireless photography", *Consumer Electronics, IEEE Transactions*, vol. 51, pp.240-244, Feb. 2005.

[5] P. Bigioi, E. Steinberg, G. Susanu, and A. Pososion, "Picture Transfer Protocol – the answer to Digital Camera Connectivity", *Proc. of the 9th Intl. Conference on Optimization of Electrical and Electronic Equipments*, vol. 3, pp. 217 -226, May 2004.

[6] P. Bigioi, G. Susanu, P. Corcoran, and I. Mocanu, "Digital camera connectivity solutions using the picture transfer protocol (PTP)", *Consumer Electronics, IEEE Transactions* vol. 48, Issue 3, pp. 417 – 427, Aug. 2002.

[7] P. Bigioi, P. Corcoran, and G. Susanu, "Digital imaging services using PTP ", *Consumer Electronics, 2002. ICCE. 2002 Digest of Technical Papers*, pp. 54-55, June 2002.

[8] Camera & Imaging Products Association (CIPA), "CIPA-DC-001-2003, digital PhotoSolutions for Imaging Devices", http://www.cipa.jp/english/pictbridge/, Japan 2003.

[9] MTP Media Transfer Protocol Specification, *Microsoft website*, http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnwmt/html/mtp_spec.asp

[10] Internet Engineering Task Force, "A DNS Resource Record for specifying the location of services (DNS SRV)", *http://www.ietf.org/rfc/rfc2782.txt*, February 2000.

[11] Internet Engineering Task Force, "Simple Service Discovery Protocol 1.0", *http://quimby.gnus.org/internet-drafts/draft-cai-ssdp-v1-03.txt* , October 1999.

[12] http://www.upnp.org

[13] http://www.upnp.org/download/draft_cai_ssdp_v1_03.txt

[14] http://www.upnp.org/download/UPnPDA10_20000613.htm

Peter Corcoran received the BAI (Electronic Engineering) and BA (Math's) degrees from Trinity College Dublin in 1984. He continued his studies at TCD and was awarded a Ph.D. in Electronic Engineering for research work in the field of Dielectric Liquids. In 1986 he was appointed to a lectureship in Electronic Engineering at NUI, Galway and is currently Vice-Dean of Engineering. His research interests include microprocessor applications, home networking, digital imaging and wireless networking technologies. He is a member of IEEE.

Ilariu Raducan received his B.Eng. degree in Electronic Engineering from the Faculty of ElectroMechanical Engineering at the University of Craiova, Romania in 1994. He is currently completing an M. Eng. Sc. degree in the Consumer Electronics Research Group at the National University of Ireland, Galway. His research interests include embedded systems design, communication network protocols, and wireless data systems and their applications.

Petronel Bigioi received his B.S. degree in Electronic Engineering from"Transilvania" University from Brasov, Romania, in 1997. At the same university he received in 1998 M.S. degree in Electronic Design Automation. He received a M.S. degree in electronic engineering at National University of Ireland, Galway in 2000. Currently he is lecturing networks and communication systems at National University of Ireland, Galway. His research interests include VLSI design, communication network protocols, digital imaging and embedded systems.

Eran Steinberg received his BSc. in mathematics from HU, Jerusalem, Israel. He received a MSc. in Imaging Science from RIT, Rochester NY. With a vast experience of over 15 years in industry, project leader of ISO/WG18/TC42/IT10 - 15740, 8 granted patents and over 20 pending patents, he is currently CTO of FotoNation. His research interests include digital image processing and connectivity.