IrDA Infrared Communications:
An Overview

By

Patrick J. Megowan
David W. Suvak
Charles D. Knutson
Counterpoint Systems Foundry, Inc.
{patm, davesu, knutson}@countersys.com

Biographies:

Patrick Megowan and David Suvak have been involved with the Infrared Data Association since the beginning. At Hewlett-Packard they created much of the pioneering HP/Microsoft Windows 95 IrDA solution. Pat and Dave subsequently founded Counterpoint Systems Foundry, Inc., to make IrDA software solutions for embedded systems and non-PC platforms. Counterpoint products are available for major RTOSs and in stand-alone, DOS, and Windows configurations. Counterpoint is heavily involved in the ongoing development of standards and technologies. Dr. Charles Knutson is Vice President of Research and Development at Counterpoint Systems Foundry.

1.0 Introduction

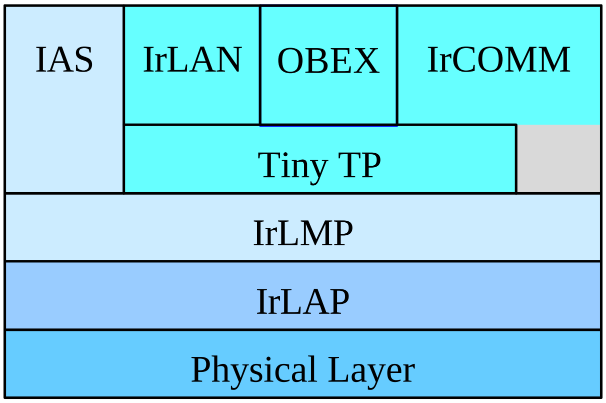
As infrared data communications, based on standards from the Infrared Data Association (IrDA), become widely available on personal computers and peripherals, a timely opportunity exists for effective and inexpensive short range wireless communications on embedded systems and devices of all types. The IrDA standards were developed rapidly (compared to most standards organizations), and information on the IrDA protocols has not yet reached every corner of the embedded systems universe. This paper gives an overview of the IrDA protocols with comments on their use in embedded environments.

The Infrared Data Association (IrDA) is an industry-based group of over 150 companies that have developed communication standards especially suited for low cost, short range, cross-platform, point-to-point communications at a wide range of speeds. These standards have been implemented on various computer platforms and more recently have become available for many embedded applications. Because of their wide acceptance, the IrDA specifications are now on an accelerated track for adoption as ISO standards.

1.1 What is an IrDA Protocol Stack?

Communications protocols deal with many issues, and so are generally broken into layers, each of which deals with a manageable set of responsibilities and supplies needed capabilities to the layers above and below. When you place the layers on top of each other, you get what is called a protocol *stack*, rather like a stack of pancakes or a stack of plates. An IrDA protocol stack is the layered set of protocols particularly aimed at point-to-point infrared communications and the applications needed in that environment.

Below is a picture of the IrDA protocol layers. This layering will serve as the overall structure for much of the remaining discussion.



The layers within this stack can be divided into two groups—required and optional protocols.

1.2 Required IrDA Protocols

The required layers of an IrDA protocol stack are in unshaded boxes in the diagram above and include the following:

- Physical Layer: Specifies optical characteristics, encoding of data, and framing for various speeds.
- IrLAP: Link Access Protocol. Establishes the basic reliable connection.
- IrLMP: Link Management Protocol. Multiplexes services and applications on the LAP connection.
- IAS: Information Access Service. Provides a “yellow pages” of services on a device.

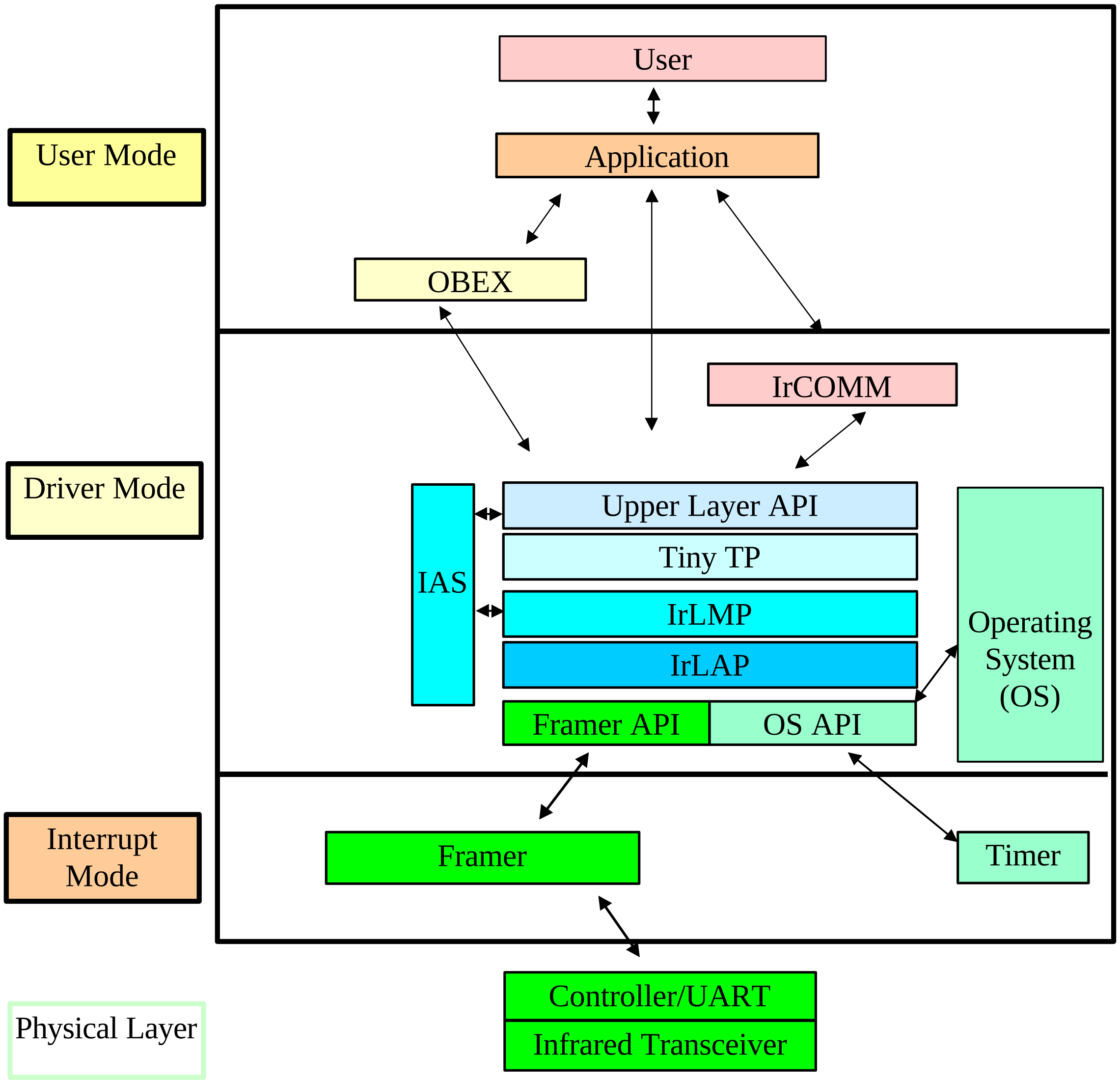
1.3 Optional Protocols

The optional protocols are shown above in shaded boxes. The use of the optional layers depends upon the particular application. The optional protocols are:

- TinyTP: Tiny Transport Protocol. Adds per-channel flow control to keep things moving smoothly. This is a very important function and *is required* in many cases.
- IrOBEX: The Object Exchange protocol. Easy transfer of files and other data objects
- IrCOMM: Serial and Parallel Port emulation, enabling existing apps that use serial and parallel communications to use IR without change.

- IrLAN: Local Area Network access, enabling walk-up IR LAN access for laptops and other devices.

When the stack layers shown above are integrated into an embedded system, the picture may look more like the following:



2.0 Physical Layer and the Framer

2.1 Framer Overview

The Physical layer includes the optical transceiver, and deals with shaping and other characteristics of infrared signals including the encoding of data bits, and some framing data such as begin and end of frame flags (BOFs and EOFs) and cyclic redundancy checks (CRCs). This layer must be at least partially implemented in hardware, but in some cases is handled entirely by hardware.

In order to isolate the remainder of the stack from the ever-changing hardware layer, a software layer called the *framer* is created. Its primary responsibility is to accept incoming frames from the hardware and present them to the Link Access Protocol layer (IrLAP). This includes accepting outgoing frames and doing whatever is necessary to send them. In addition, the framer is responsible for changing hardware speeds at the bidding of the IrLAP layer, using whatever magic incantations the hardware designer invented for that purpose (these signals have not yet been standardized).

3.0 IrLAP - Link Access Protocol

3.1 IrLAP Overview

Immediately above the framer we encounter the IrLAP layer, also known as the Link Access Protocol, or LAP for short. IrLAP is a required IrDA protocol corresponding to OSI layer 2 (data link protocol). It is based on High-Level Data Link Control (HDLC) and Synchronous Data Link Control (SDLC) with extensions for some unique characteristics of infrared communications.

IrLAP provides reliable data transfer using the following mechanisms:

- Retransmission.
- Low-level flow control. (TinyTP provides high-level flow control and should almost always be used in place of IrLAP flow control.)
- Error detection.

By dealing with reliable data transfer at a low level, upper layers are free from this concern and can be assured that their data will be delivered (or at least that they will be informed if it was not). Data delivery might fail if the beam path were blocked. For instance, someone could put a coffee cup in the path of the infrared beam. IrLAP alerts the upper layer so that higher-level layers can deal with the problem appropriately. As an example, an application sitting on the stack could be alerted of an interruption in data flow, allowing it to alert the user through some interface. The user could then potentially remedy the problem (by moving the coffee cup) without dropping the connection or losing the data transferred to that point.

3.2 Environmental Characteristics

Several environmental factors influenced the development of the IrLAP layer. These include the following:

- *Point-to-point*. Connections are one-to-one, such as camera to PC or data collector to printer. The range is typically zero to one meter, although extended range up to 10 meters or more is under development. This is not like a local area network (many-to-many) protocol.
- *Half-duplex*. Infrared light, and therefore data, is sent in one direction at a time. However, the link changes directions frequently and can simulate full duplex in cases where timing is not extremely sensitive.
- *Narrow infrared cone*. The infrared transmission is directional within a 15 degree half angle in order to minimize interference with surrounding devices.
- *Hidden Nodes*. Other IR devices approaching an existing connection may not be immediately aware of the connection if they approach from behind the current transmitter. They must wait and see if the link turns around before stepping in.
- *Interference*. IrLAP must overcome interference from fluorescent lights, other IR devices, sunlight, moonbeams, and so forth.
- *No collision detection*. The design of the hardware is such that collisions are not detected, so the software must handle cases where collisions cause lost data with methods such as random back off.

3.3 Roles within a LAP connection

The two parties to a LAP connection have a master-slave relationship with differing responsibilities (and resulting code complexity). The IrDA terms for this are Primary (master) and Secondary (slave).

Primary Station:

- Sends Command frames—initiates connections and transfers.
- Responsible for organization and control of data flow.
- Deals with unrecoverable data link errors.
- Typical primary devices include PCs, PDAs, cameras, and anything that needs to print (printers are currently all secondaries).

Secondary Station:

- Sends Response frames—only speaks when spoken to.

- Typical secondary devices are printers and other peripherals, and resource-constrained devices (secondaries are smaller and less complex).

In any connection one device must play the primary role. The other device must play the secondary role, but its protocol stack may be either a secondary or another primary—most primaries can play a secondary role. Once started, the two sides take turns talking with the primary leading off. No side can talk for more than 500 milliseconds at a time before allowing the other side a chance to talk (even if just to say it has nothing to send for the moment).

Note that the issue of master versus slave becomes much less obvious at the higher protocol layers—*once two devices are connected*, an application on a secondary (slave) can initiate an operation just as easily as an application on the primary side.

3.4 Modes

IrLAP is built around two modes of operation, corresponding to whether or not a connection exists.

3.4.1 Normal Disconnect Mode (NDM)

NDM is also known as contention state, and is the default state of disconnected devices. In this mode a device must observe a set of media access rules. Of utmost importance, a device in NDM must check whether other transmissions are occurring (a condition known as *media busy*) before transmitting. This is accomplished by listening for activity. If no activity is detected for greater than 500 milliseconds (the maximum time for the link to turn around), the media is considered to be available for establishment of a connection.

A great ease-of-use feature is provided by the NDM communications rules. A classic problem is getting both sides of the link configured with the same communications parameters—frequently users get completely stuck. This can be particularly difficult on embedded devices that don’t have a user interface for setting or reviewing communications parameters. This problem is completely absent with IrDA solutions—all NDM communications use the following link parameters: ASYNC, 9600 bps, 8 bits, no parity. During the connection process, the two sides exchange capability information, and subsequently shift to the best parameters supportable by both sides.

3.4.2 Normal Response Mode (NRM)

NRM is the mode of operation for connected devices. Once both sides are talking using the best possible communication parameters (established during NDM), higher stack layers use normal command and response frames to exchange information.

3.5 IrLAP Frame Format

The basic IrLAP frame format is as follows:

Address	Control	Information

While there are too many details about frame formats to discuss here, it is worth noting that the Address and Control fields require only two bytes total—the IrDA protocols add very little overhead to the user data.

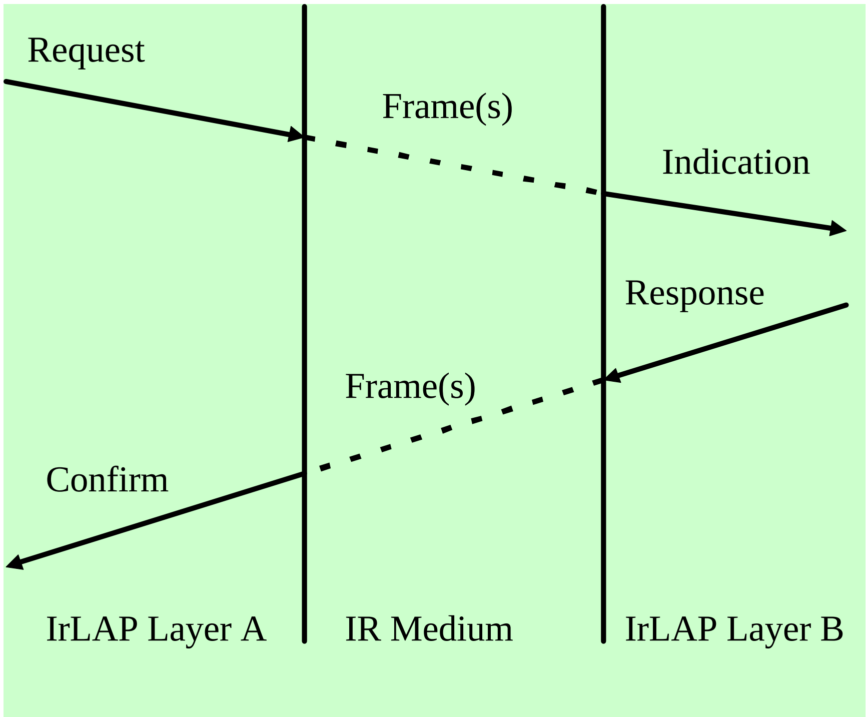
3.6 IrLAP Frame Wrappers

Before sending, a frame is wrapped with framing information. Three different frame wrappers are used by IrLAP, depending upon the speed of the connection.

- Asynchronous (ASYNC) Framing: 9600 bps - 115.2 kbps
- Synchronous (SYNC) HDLC Framing: 576 kbps and 1.152 Mbps
- Synchronous 4 PPM Framing: 4 Mbps

3.7 Service Primitive Diagram

IrLAP operations are described in the specification using *service primitives*. You can think of the service primitive as a *conceptual model* of an API for an operation that IrLAP performs (actual APIs for IrLAP services are completely up to the developer). This diagram illustrates an operation: it starts with a service *request*, travels across the link as a frame, is reported as an *indication* (frequently an up-call) on the receiving side; the receiver then formulates a *response* which travels back as a frame, finally resulting in a *confirm* (often an up-call) to the original requester.



3.8 IrLAP Services

A number of services are defined in the IrLAP specification. Not all services are necessary for all devices, and the IrLAP specification (along with the IrDA Lite standard) describes the minimum requirements. The most important services include the following:

- Device Discovery: Explores the nearby IR-space to see who is present and get some hint as to what they can do.
- Connect: Chooses a specific partner, negotiates the best possible communication parameters supported by both sides, and connects.
- Send Data: The whole reason for all this effort—used by higher layer protocols for almost all of their work.
- Disconnect: Closes down and returns to the NDM state, ready for a new connection.

4.0 IrLMP - Link Management Protocol

4.1 IrLMP Overview

The IrLMP layer depends upon the reliable connection and negotiated performance provided by the IrLAP layer. IrLMP is a required IrDA layer, and provides the following functionality:

- Multiplexing: LMP allows multiple IrLMP clients to run over a single IrLAP link.
- Higher level discovery, consisting of:
    - Address conflict resolution on IrLAP Discovery. Handles the case of multiple devices with the same IrLAP address by telling them to generate new addresses
    - Information Access Service (IAS). A “yellow pages” describing the services available on a device.

4.2 IrLMP Terminology

In order to have multiple IrLMP connections on a single IrLAP connection, there must be some higher level addressing scheme. The following terminology is used to describe this addressing:

- LSAP (Logical Service Access Point): The point of access to a service or application within IrLMP (for example, a printing service). It is referenced with a simple one byte number, the LSAP-SEL (described next).
- LSAP-SEL (LSAP Selector): A one byte number that corresponds to an LSAP. Think of this as the *address* of a service within the LMP multiplexor. This byte is broken into ranges—0x00 is the IAS server, 0x01 through 0x6F are legal LMP connections, 0x70 are for connectionless services (not discussed in this paper), and the rest are reserved for future use.

Given the limited number of LSAP-SEL values, services are not assigned fixed “port addresses” as in TCP/IP. Instead, services have fixed published names, and the LMP IAS (yellow pages) is used to look up the LSAP-SEL for a desired service.

4.3 IrLMP Services

Here are the services defined in the IrLMP specification. Not all services are necessary in all devices, and the specification (along with the IrDA Lite standard) describes the minimum requirements. Notice that this set is identical to the set listed in the IrLAP section above—it is a common feature of protocol stacks for operations to propagate upward like this, with each layer adding its particular contribution.

- Device Discovery: Finds out additional information about devices in the IR space.
- Connect: Establishes a connection between a pair of services at the LMP level.
- Data: Sends the data back and forth.
- Disconnect: Closes this LMP connection. Note that this does not necessarily

4.4 Frame Format

The IrLMP layer adds the following 2 bytes of information to frames in order to perform its basic operations:

7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
C	DLSAP-SEL							r	SLSAP-SEL						

- C: Distinguishes between control and data frames.
- r: Reserved.
- DLSAP-SEL: LSAP-SEL (service address) of the destination of the current frame.
- SLSAP-SEL: LSAP-SEL for the sender of the current frame

5.0 IAS - The Information Access Service

5.1 IAS - The Yellow Pages for Services and Applications

The IAS, or Information Access Service, acts as the “yellow pages” for a device. All of the services/applications available for incoming connections must have entries in the IAS which can be used to determine the service address (LSAP-SEL). The IAS can also be queried for additional information about services.

A full IAS implementation consists of client and server components. The *client* is the component that makes inquiries about services on the other device using the Information Access Protocol (IAP, used only within the IAS). The *server* is the component that knows how to respond to inquiries from an IAS client. The server uses an information base of objects supplied by the local services/applications. In fixed-purpose embedded systems this may be a hard-coded collections of objects, while in a PDA there may be APIs for registering and de-registering services. Note that devices which never initiate LMP connections might include an IAS server only.

5.2 IAS Information Model

The IAS Information Base is a collection of objects that describe the services available for incoming connections. The information base is used by the IAS server to respond to incoming IAS queries.

Information base objects consist of a *class name* and one or more *attributes*. They are quite similar to entries in the yellow pages of a phone book. The class name is equivalent to the business name in the phone book—it is the official published name of the service or application. IAS clients will inquire about a service using this name. The attributes contain information analogous to phone number, address, and other characteristics of the business. The one essential attribute for every entry is the LSAP-SEL (or service address), which is required in order to make a LMP connection to the service.

An IAS object is made of the following pieces:

- Class Name (up to 60 octets).
- Named attributes (up to 60 octet names):
    - Up to 256 attributes.
    - Attributes value types:
        - User string (up to 256 octets)
        - Octet Sequence (up to 1024 octets)
        - Signed Integer (32-bit).

5.3 Getting information using the IAS

There are a number of IAS operations defined in the IrLMP standard, but the most used and only required one is called *GetValueByClass*. The inquiring party gives the class name (for example, Printer) and the name of the attribute it wants (for example, the LSAP-SEL), and receives a response that consist of one or more answers (for example, the LSAP-SELs for any printing services in the responding party's information base) or an indication that the service or attribute does not exist.

IAS Query Arguments:

- Class Name Length (1 octet).
- Class Name (*Length* octets).
- Attribute Name Length (1 octet).
- Attribute Name (*Length* octets).

Results:

- Return Code:
    - 0: Success, results follow.
    - 1: No such class, no results follow.
    - 2: No such attribute, no results follow.

If the result code indicates success, the call returns the following information:

- List Length (2 octets).
- List of results:
    - Object Identifier (2 octets).
    - Attribute value (based on attribute type).

6.0 TinyTP - the Tiny Transport Protocol

6.1 TinyTP Overview

To put TinyTP in its proper perspective, it would help to review the layers covered so far:

- *Physical layer* defines the hardware requirements and low-level framing of the data.
- *IrLAP* provides reliable, sequenced data, and trouble-free connection at agreed upon parameters with automatic negotiation to best common parameters.
- *IrLMP* provides multiplexing of services onto the LAP connection and the IAS “yellow pages” of services available for incoming connection.

TinyTP (TTP for short) is an optional IrDA layer, although it is so important that it should generally be considered a required layer (except in the case of current printing solutions). TTP provides two functions:

- Flow control on a per-LMP-connection (per-channel) basis.
- SAR (segmentation and reassembly).

TTP adds one byte of information to each IrLMP packet to perform its task.

6.2 Tiny TP Flow Control

Per-channel flow control is currently the most important use of TinyTP. You may recall that IrLAP offers flow control and wonder why another flow control mechanism is needed. To illustrate the need, suppose that a LAP connection is established, and two LMP connections are made on top of the LAP connection using the LMP multiplexing capability. If one side turns LAP flow control on, the flow of data on the LAP connection (which carries all the LMP connections) is completely cut in that direction, and the other side cannot get the data it wants until LAP flow control is turned off. The work of the second side may be seriously disrupted (especially if timers are involved).

If flow control is applied on a per-LMP-connection basis using TinyTP (or other mechanisms), then one side can stop to digest information without negatively affecting the other side.

6.3 How TTP flow control works

TTP is a credit-based flow control scheme. It works as follows:

- At connection, some credit is extended by each side. One credit corresponds to permission to send one LMP packet. If you send a credit, you must be able to accept a maximum sized packet. You can see that the number of credits you extend depends entirely on how much buffer space you have. As long as you have buffers, you can send anywhere from 1 to 127 credits.
- Sending data causes credit to be used up (1 unit of credit per packet sent).
- Periodically, the receiver issues more credit. This “credit policy” is entirely at the receiver’s discretion, but the policy can make a big difference in the performance of the link. If the sender is constantly running out of credit and having to wait for more, throughput will suffer.
- If a sender has no credit, no data movement can occur, but…
- A credit-only packet can always be sent—it is not subject to flow control.

Although the above description talks about the sender and receiver as if those roles are fixed, it is common for both sides of a LMP connection to send and receive, hence both sides will be issuing and using credit. Note that the credit byte normally travels as part of a LMP data packet, so LMP packets are not needed just for sending credit as long as there is data to send and credit to send with. Obviously a little head scratching is in order to set up an efficient credit policy.

6.4 Segmentation and Reassembly

The other TTP function is called SAR (segmentation and re-assembly). The basic idea is that TTP breaks large data into pieces (segmentation), and puts it back together on the other side (re-assembly). The entire piece of data being chopped up and re-constituted is called an SDU, or Service Data Unit, and the maximum SDU size is negotiated when the TTP/LMP connection is first made.

6.5 Tiny TP Service Primitives

As with the IrLAP and IrLMP layers, TTP operations are characterized as service primitives:

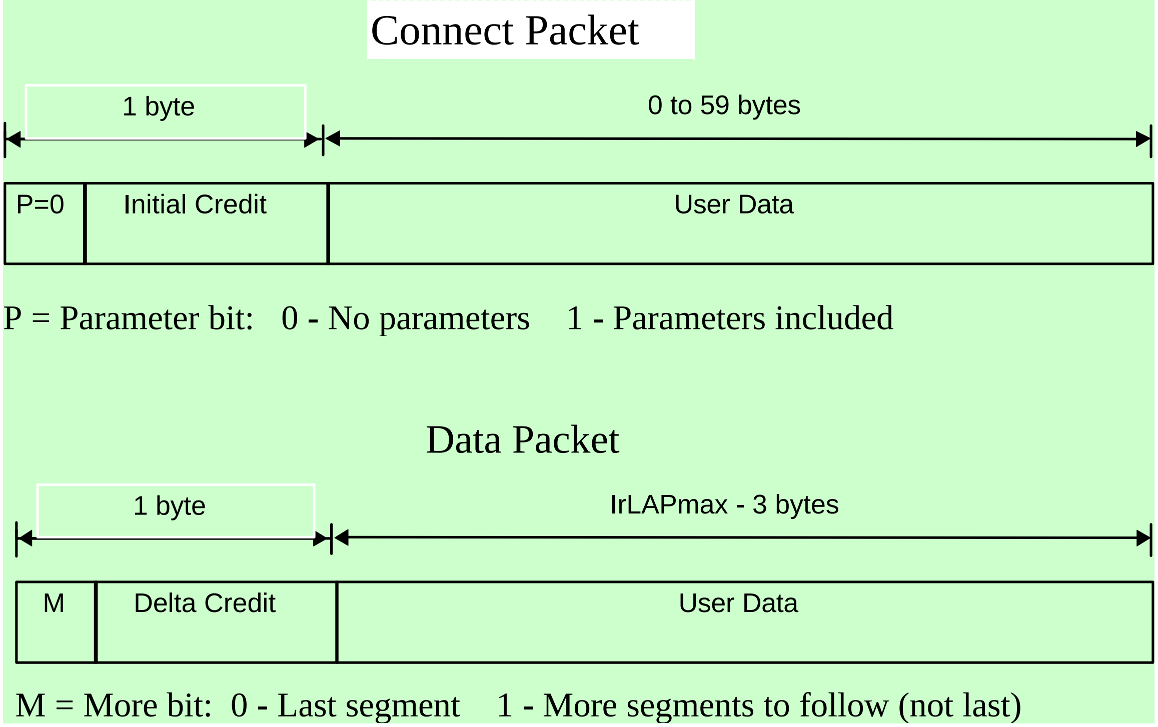
- Connect: Negotiate maximum SDU size.
- Disconnect.
- Data: Reliable sequenced data.
- Local Flow Control: Stop data delivery.

- Udata: Unreliable, unsequenced (pass through to IrLMP).

TTP service primitives focus on the core of the LMP primitives—connect, send, and disconnect, adding a means to exert flow control.

6.6 TTP Frame Formats

The two frame formats used by TTP are the *connect packet* (carried with the IrLMP connect packet, hence the limited data length), and the *data packet*, carried with IrLMP data packets.



P = Parameter bit: 0 - No parameters 1 - Parameters included

M = More bit: 0 - Last segment 1 - More segments to follow (not last)

7.0 IrDA Lite - IrDA Gets Small

7.1 IrDA Lite Overview

Up until this section, we have discussed layers of the IrDA protocol stack. This section briefly describes a specification which does not create a new layer, but which modifies layers already described.

The IrDA Lite specification describes a set of design and implementation strategies that together yield the smallest possible implementation that will still perform connection-oriented communications with a full IrDA implementation.

Naturally, the ultimate size of the stack will depend heavily on the hardware, the software tools available, and the experience and skills of the development team. However, our experience suggests that primaries under 10 Kbytes code size and secondaries under 5K are feasible on common CISC processors used in embedded systems. RAM usage can be as low as a few hundred bytes, most of which is needed to buffer incoming frames.

7.2 An All or Nothing Deal?

The IrDA Lite specification is composed of a number of strategies. A developer may choose to implement all of them, or only the ones appropriate to a particular device. For instance, some of the strategies severely limit the performance of the stack:

- Speeds restricted to 9600 bps.
- LAP packet size restricted 64 bytes.

While the most severely constrained devices (watches, for instance) may both need and tolerate these constraints, many devices (such as digital cameras) need high performance. Some strategies do not affect performance at all, so a judicious choice is in order to balance the needs for performance, functionality, and size.

By way of example, Counterpoint Systems Foundry offers three performance/size points in its IrLite products: Compact (implements all possible IrDA Lite strategies); Extended (allows speeds up to 4 Mbps and large packet sizes, but limits window size to one); and Performance (Extended version plus window size up to seven—implements only IrDA Lite strategies that do not affect throughput). Other combinations of features are possible, but this illustrates that there is a substantial range of possible implementations under IrDA Lite.

7.3 A Key element of IrDA Lite

While a description of most of the IrDA Lite strategies is beyond the scope of this paper, it is worth noting the following key strategy and its effect on the decision to use an IrDA Lite based protocol implementation.

In a non-IrDA Lite setting, applications issue an IrLMP disconnect when they are done, and IrLMP closes the LAP connection when all the LMP connections are done. In contrast, under IrDA Lite applications use the IrLAP disconnect operation when they are done. Suppose we have two applications communicating on their respective LMP channels with counterparts on another device. If one side finishes up and issues an IrLAP disconnect, the other side's connection will terminate without warning, a potentially disastrous scenario.

Many embedded systems have a focused purpose and will find a single communication channel adequate. If multiple applications desire separate channels *at the same time*, things will still work as long as they are aware of each other and employ a "last one to leave turns out the light" approach to the IrLAP disconnect. Alternatively, it may be much easier for all applications to send all their data through an IrOBEX implementation (discussed in the next section), and let OBEX take care of making and breaking connections for everyone, as well as handling all the intricacies and contingencies that occur in any communications task.

8.0 IrOBEX - Object Exchange Protocol

8.1 IrOBEX overview

IrOBEX is an optional application layer protocol designed to enable systems of all sizes and types to exchange a wide variety of data and commands in a resource-sensitive standardized fashion. It addresses one of the most common applications on either PCs or embedded systems: take an arbitrary data object (a file, for instance), and send it to whoever the infrared device is pointing to. It also provides some tools to enable the object to be recognized and handled intelligently on the receiving side.

The potential range of objects is wide, encompassing not only traditional files, but also pages, phone messages, digital images, electronic business cards, database records, hand-held instrument results, or diagnostics and programming. The common thread is that the application doesn't need or want to get involved in managing connections or dealing with the communications process at all. Just take the object and ship it to the other side with the least fuss possible. It is very similar to the role that HTTP serves in the Internet protocol suite, although HTTP is very "pull"-oriented in its fundamental design, while OBEX is more evenly balanced.

8.2 Characteristics of IrOBEX protocol

OBEX was created to "package" an IrDA communications transaction as completely as possible and thereby dramatically simplify the development of communications-enabled applications. It was further designed to meet the following criteria:

- Simple: Supports most-needed operations/applications.
- Compact: Under 1K code on small system.

- Flexible: Supports data handling for both industry standard and custom types.
- Extensible and Debug-able.
- Works on IrDA, but is transport independent.

8.3 Components of IrOBEX protocol

The OBEX standard consists of the following pieces:

- Session model: The rules of conversation governing the exchange of objects. Includes optional negotiation during connection, a set of operations such as Put and Get. Allows terminating the transfer of an object without closing the connection. Supports graceful close of a connection.
- Object model: Provides a flexible and extensible representation for objects and information describing the object.
- Guidelines for use and extension:
    - Defining new session operations.
    - Defining new object types
- IAS entry for a default OBEX server, and suggestions for its capability.

9.0 IrCOMM - Serial and Parallel Port Emulation

9.1 IrCOMM Overview

When the IrDA standards were developed, there was a strong desire to allow existing PC applications that use serial and parallel ports to operate via infrared without change. These applications, collectively known as “legacy applications”, included printing, file transfer applications such as LapLink or Carbon Copy, and modem communications.

However, IrDA infrared communications differs significantly from serial and parallel communications. For instance, both serial and parallel cables have individual circuits over which signals can be sent independently and concurrently. By contrast, infrared has a single beam of light, and all information must be fitted into LMP or higher layer packets in a serial stream.

The IrCOMM standard was developed to solve these problems and allow legacy applications to be used over infrared with a minimum of hassle. The key feature of IrCOMM is the definition of a so-called *control channel* to carry the non-data circuit information. In the stack picture, IrCOMM rests on top of IrLMP and TinyTP.

IrCOMM is an optional IrDA protocol that applies only to certain applications. In general, new applications are better served if they avoid IrCOMM and use other IrDA applications protocols such as IrOBEX, IrLAN, or TinyTP directly. This is because IrCOMM masks some of the useful features built into the lower protocols. After all, its job is to make IrDA look like serial and parallel media that do not have handy features like automatic negotiation of best common parameters and a “yellow pages” of available services.

9.2 IrCOMM Service Types

Because different applications use the non-data circuits of serial and parallel communications to varying degrees, four service types are defined in IrCOMM:

- 3-Wire Raw (Parallel and Serial Emulation): Sends data only, no non-data circuit information and hence no control channel. Runs directly on IrLMP.
- 3-Wire (Parallel and Serial Emulation): Minimal use of control channel. Uses Tiny TP.
- 9-Wire (Serial emulation only): Uses control channel for status of standard RS-232 non-data circuits. Uses Tiny TP.
- Centronics (Parallel emulation only): Uses control channel for status of Centronics non-data circuits. Uses Tiny TP.

10.0 IrLAN - LAN access

10.1 IrLAN Overview

The final optional protocol discussed is IrLAN. It is mentioned only briefly because it is not an approved standard at this time, nor is its use widespread in the world of embedded systems—it primarily serves as an extremely convenient connection between portable PCs and office LANs.

IrLAN offers three models of operation:

- Enable a computer to attach to a LAN via an Access Point Device (sometime called an IR LAN Adapter). The Hewlett Packard NetBeam IR is an example of this type of device.
- Enable two computers to communicate as though attached on a LAN—in effect an instant LAN between a pair of machines, with access to the other machines’ directories and other LAN capabilities.
- Enable a computer to attach to a LAN through a second computer already attached.

11.0 Summary

IrDA Protocol stack implementations are available today on Windows 95, Window

3.x, Macintosh, a few PDAs, peripherals and adapters from HP, ESI and others. Stacks are also available in kit form for embedded applications, and are rapidly approaching on major embedded operating systems such as pSOS, OS/9, VxWorks, GeoWorks, Windows CE, Itron, and others.

Features like automatic selection of compatible communication parameters and service “yellow pages” make the IrDA protocols well-suited to embedded devices, even in consumer markets where the device must communicate simply (like a TV remote control) to gain wide-spread acceptance.

The IrDA standards documents and additional information about the Infrared Data Association are available at www.irda.org. The IrDA web site also includes links to suppliers of hardware and software. Information about Counterpoint Systems Foundry, Inc., can be obtained at www.countersys.com.

The IrDA meets quarterly (most often in the California Bay Area) to discuss ongoing standards work, show off and discuss new products and technologies, and bring together individuals and companies with common interests.