

Multiplexed Serial Wireless Connectivity for Palmtop Computers

Ibrahim Korpeoglu, Pravin Bhagwat, Chatschik Bisdikian, Mahmoud Naghshineh*

*Computer Science Department
University of Maryland
College Park, MD 20742

IBM T. J. Watson Research Center
P.O.Box 218
Yorktown Heights, NY 10598

ABSTRACT

Palmtop computers which consume very little power, have very small size, and are much cheaper compared to their laptop counterparts are proliferating in the market. However, there exist no satisfactory wireless connectivity solution in terms of cost and power consumption to connect these devices to the network. This paper describes a cost-effective, low-power wireless connectivity solution for these devices and presents the design and implementation of how multiple such devices are connected to an access point over a shared wireless link.

1 Introduction

Palmtop computers are mostly used in disconnected mode for storing address book information, recording appointments, taking notes, etc. Network-based interactive applications have not yet proliferated. The most commonly used networked application is (network) hotsync, which synchronizes the data in a PDA with a desktop computer through an RS-232 serial cable or through modem connected telephone lines (figure 1-a,b). The main reason for unpopularity of networked applications is the lack of good wireless connectivity solutions. Existing wireless solutions are not suitable for networking palmtop devices.

The networking requirements of PDAs differ from those of laptop and desktop computers. Unlike laptop computers, PDAs are designed to operate within strict cost, space, and power budget. Therefore, any communication solution for PDAs need to have similar properties in terms of cost, battery efficiency and form factor. Secondly, since most PDAs only support RS-232 serial interface to communicate, traditional PCMCIA based wireless LAN cards cannot be used.

Although various indoor and outdoor existing wireless network technologies can be adapted for enabling wireless access for PDAs, none of these wireless technologies provide a cost effective solution. Wide area solutions such as CDPD, ARDIS, RAM are not suitable for indoor use. Metricom's Ricochet wireless modem design is an attractive alternative. However, Ricochet is intended to be a wide area wireless service, not a plug-n-play, build-your-own-kind of a wireless system. It is fee-based, therefore is not economically attractive, and does not provide a low-cost method of connecting a PDA directly to a LAN. A pair of Ricochet modems can be used to form a wireless RS-232 link, which can be used to connect a PDA directly to a PPP server (figure 1-c). This solution, however, has

three limitations: multiple users cannot access the network simultaneously; roaming is not supported; and the configuration is expensive. A Ricochet modem also consumes more power than a PDA.

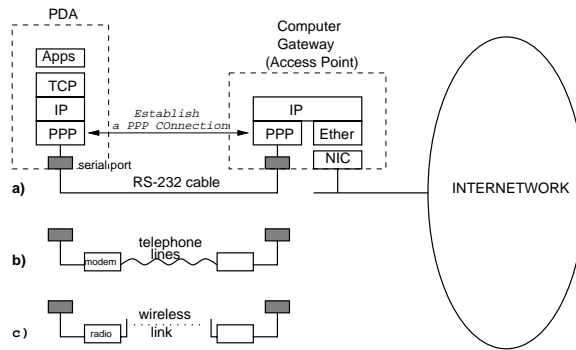


Figure 1: Several current ways to connect a PDA to an internetwork over point-to-point links.

To address the above issues, we designed a new wireless access network (called BlueSky) for PDAs considering low cost, low power consumption, and light-weight protocols as the primary objectives. In this paper, we will first briefly describe the BlueSky architecture, the entities and the protocols. Then we will present our design and implementation to enable multi-access capability for PDAs to share a single wireless link that connects the devices to an access point.

2 BlueSky Approach to Wireless Connectivity

The BlueSky system consists of wireless *BlueSky attachments* that are connected to PDAs and provide short range wireless access to the wired network backbone through the *BlueSky access points* [1]. Multiple PDAs can share a wireless link to an access point using the BlueSky wireless MAC protocol (see figure 2).

A power-conscious, polling based, asymmetric MAC protocol is designed for the wireless link [2]. The portion of the MAC protocol that is running on an access point, called AP-MAC, is more intelligent and coordinates the access to the medium by sending commands (polls) to the PDAs in the range. It includes a scheduler component which schedules MAC-level events such as transport-payload, poll-and-receive-payload, invite-stations, etc. The corresponding MAC protocol that is running on a PDA attachment, called RS-MAC (remote station MAC), just responds to the commands sent from the AP-MAC and is much simpler, requires much less memory and processing power, and has smaller code size than the AP-MAC. The protocol supports sleep modes to conserve energy.

The wireless attachment has been designed as an external attachment so that it can be plugged into any PDA, laptop, or serial port enabled device. It connects to the serial port of a device. It has a radio transceiver component and a microcontroller component. These two are also connected internally through a serial line. Since we did not want to modify the PDA hardware and software to implement the RS-MAC protocol, the microcontroller part is needed to implement it. The RS-MAC layer packetizes the bytes received from a PDA through the serial port and sends them to an access point inside MAC frames.

Since the cost of the radio transceiver is an important factor in the overall cost of the BlueSky attachment and access point, we chose a cost effective radio transceiver which runs in ISM 915 MHz band and uses simple FSK modulation similar to what is used in cordless phones [3]. The attachment and access point can select one of the ten available frequency channels. The link bandwidth is 150

Kbps and the effective range is between 20-30 m depending on the interference and multi-path effects. The bandwidth can be shared among approximately 20 mobile devices.

An access point consists of a PC-104 board (with a 386 CPU on it), a radio transceiver, and an Ethernet card to connect to the internetwork. The AP-MAC protocol is implemented as part of the other access point software, hence a separate microcontroller is not needed. The radio transceiver is connected to the access point board through a serial port. (see figure 3). The byte streams received by the radio unit is given to the AP-MAC protocol through the serial port.

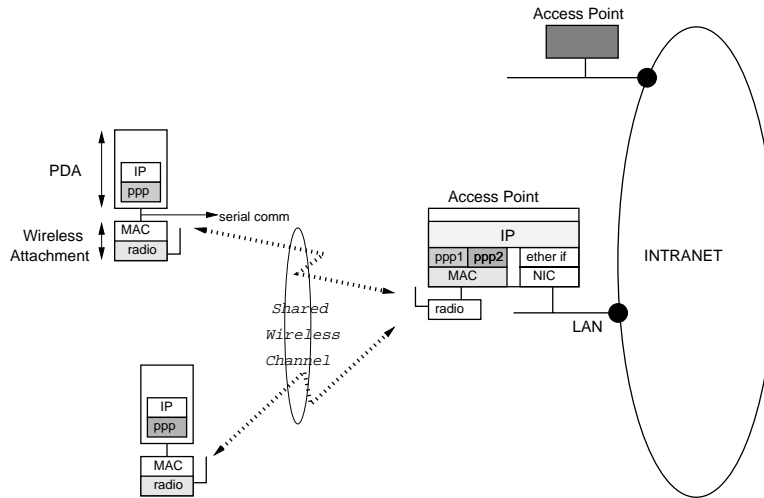


Figure 2: BlueSky architecture

A Unix operating system BSDI/OS 3.0 runs on an access point to implement the AP-MAC protocol, the PPP protocol [4], and the IP stack [5]. The PPP protocol connects a PDA to the access point, and the IP stack connects the access point to the rest of the internetwork. Multiple PDAs can establish PPP connections to the same access point at the same time. PPP protocol exports a packet-oriented network interface to the IP layer so that IP can also run over serial communication lines. There needs to be a separate PPP network interface/server running for each PDA that established a PPP connection to the access point. The challenge is putting an AP-MAC layer under these multiple PPP interfaces (see figure 2) and emulating a serial line communication over this packet-oriented MAC layer. The traffic going to multiple PDAs from multiple PPP interfaces is multiplexed at the AP-MAC layer and then transported over the shared wireless link. Similarly, the traffic coming to the access point from multiple PDAs through the coordination of the AP-MAC layer is demultiplexed to multiple PPP interfaces.

3 Multiplexing PPP Traffic at the MAC Layer

The PPP implementation in BSD Unix consists of two parts (see figure 3):

- *An asynchronous PPP driver* inside the kernel which implements the PPP framing protocol and byte stuffing algorithm. PPP driver is located under the IP layer and exports a network interface to the IP layer. A PPP packet received from the serial line may be a PPP data packet or a PPP control packet (they are distinguished by the protocol identifier field in the PPP header). A PPP data packet contains the payload for the higher network layer (IP in this case). The PPP

driver gives the payload to the IP layer after stripping off the PPP header and CRC. A PPP control packet carries PPP specific information parameters and is given to the corresponding PPP server daemon after again stripping off the PPP header and CRC.

- A *PPP (server) daemon* that runs in user space and implements the PPP control protocols. These control protocols include LCP (link control protocol), NCP (network control protocol), etc. All PPP packets generated by the PPP daemon are control packets that are used to establish, negotiate, maintain, and terminate PPP connections.

An asynchronous PPP driver usually runs on top of a serial communication port driver. In Unix, each port driver (also called terminal device driver) has a line discipline associated with it [6, 7]. A line discipline is located on top of the terminal device driver and has functions such as echoing characters back, assembling characters into lines (a line is a string of characters ending with a newline character), editing the lines, processing the flow control characters, etc. For PPP, these functionalities are not needed, and therefore a line discipline in this sense is not needed. The asynchronous PPP driver acts as a line discipline for PPP.

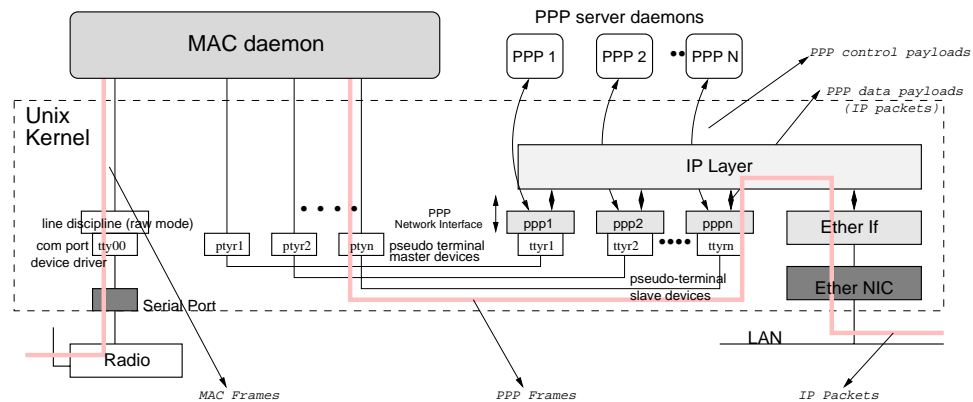


Figure 3: BlueSky access point software architecture

There is only one actual terminal device driver for a serial port, which can be used by a single PPP server. But, what we need is to be able to run multiple PPP servers and multiplex their traffic at the underlying MAC layer which then transmits the packets one-at-a-time to the corresponding PDAs through the serial port.

In order to achieve this, multiple instances of PPP servers are run on top of virtual communication ports (pseudo-terminal devices) instead of being run on top of an actual serial port hardware. In Unix, a pseudo-terminal device is a virtual device that is not associated with any physical serial port hardware, but emulates serial line communication. It consists of a pair of devices: one half is called the master and the other half is called the slave.

A process can open a pair of pseudo-terminal devices and can get two file descriptors. Anything written to the master device can be read as input from the slave device and anything written to the slave device can be read as input from the master device. Similarly, two processes can open one half of a pseudo-terminal device pair. In this case, these two processes can communicate with each other using the pseudo-terminal device pair, as if they are communicating through a serial port: they can set the line speed, etc. The slave portion of a pseudo-terminal device presents an interface to the user process that looks like an actual terminal device. It therefore has also a line discipline associated with it.

In our implementation, a PPP server/interface is run on top of a pseudo-terminal slave device. The line discipline associated with the slave device is the PPP framing protocol (see figure 3). Anything that is sent from a PPP interface to the pseudo-terminal slave is given to the respective master device. We implemented the AP-MAC protocol as a user level daemon that can read from (or write to) this master device and thereby communicate with the corresponding PPP side. Up to 64 pseudo-terminal device pairs can be opened at the same time (this number depends on the operating system and its kernel configuration). Therefore up to 64 PPP servers/interfaces can be active at the same time, each one using a different pseudo-terminal slave device. The MAC daemon can open the respective master devices and can communicate with these PPP interfaces. A PPP frame received from a PPP interface by the MAC daemon has the same format as it would have been received at an actual serial port driver. In this way, the traffic from multiple PPP interfaces are multiplexed at the MAC layer.

The MAC daemon communicates with the radio transceiver through a serial port. The bytes received from the serial port are assembled to MAC frames. The extracted payloads, which are PPP packets, are given to the respective PPP interface through a pseudo-terminal device. On the other side, the PPP packets received from a PPP interface are first queued at the MAC daemon. There is a separate queue for each PDA. The scheduler component of the MAC daemon selects the PPP packet for transmission. The selected packet is then transmitted to the respective PDA inside a MAC frame.

4 Conclusion

Palmtop computers which consume very little power, have very small form factor, and are relatively much cheaper compared to their laptop counterparts are proliferating in the market. We described a cost-effective, low-power wireless connectivity solution for these devices and presented our implementation of a multi-access scheme to connect multiple such devices to a single access point over a shared wireless link. The design and implementation of the wireless system shows how existing hardware and software pieces can be utilized to provide a very low-cost wireless connectivity solution with least modifications to the existing palmtop device hardware and software.

References

- [1] P. Bhagwat, C. Bisdikian, I. Korpeoglu, M. Naghshineh, and S. K. Tripathi, "Cordless Dialup Networking for Palmtop Computers," Tech. Rep. RC21404(96651), IBM Research, February 1999.
- [2] C. Bisdikian, P. Bhagwat, B. P. Gaucher, F. J. Janniello, I. Korpeoglu, M. Naghshineh, and P. Pandoh, "WiSAP: A Wireless Personal Access Network for Handheld Computing Devices," *IEEE Personal Communications*, December 1998.
- [3] P. Bhagwat, C. Bisdikian, I. Korpeoglu, A. Krishna, and M. Naghshineh, "System Design Issues for Low-Power, Low-Cost Short Range Wireless Networking," in *IEEE International Conference on Personal Wireless Communications (ICPWC)*, February 1999.
- [4] W. Simpson, "The Point-to-Point Protocol (PPP)," July 1994, RFC 1661.
- [5] W.R. Stevens, *TCP/IP Illustrated, Volume 1*, Addison-Wesley, 1994.
- [6] W. R. Stevens, *Unix Network Programming*, Prentice Hall, 1990.

- [7] M.K. McKusick, K. Bostic, M.J. Karels, and J.R. Quarterman, *The Design and Implementation of the 4.4BSD Operating System*, Addison-Wesley, 1996.