AIRMAIL: A Link-Layer Protocol for Wireless Networks

Ender Ayanoglu
Sanjoy Paul
Thomas F. LaPorta
Krishan K. Sabnani
Richard D. Gitlin

AT&T Bell Laboratories
Holmdel, New Jersey 07733

Abstract

This paper describes the design and performance of a link-layer protocol for indoor and outdoor wireless networks. The protocol is asymmetric to reduce the processing load at the mobile, reliability is established by a combination of automatic repeat request and forward error correction, and link-layer packets are transferred appropriately during handoffs. The protocol is named *AIRMAIL* (AsymmetrIc Reliable Mobile Access In Link-layer). The asymmetry is needed in the design because the mobile terminals have limited power and smaller processing capability than the base stations. The key ideas in the asymmetric protocol design consist of placing bulk of the intelligence in the base station as opposed to placing it symmetrically, in requiring the mobile terminal to combine several acknowledgments into a single acknowledgment to conserve power, and in designing the base stations to send periodic status messages, while making the acknowledgment from the mobile terminal event-driven. The asymmetry in the protocol design results in a one-third reduction of compiled code. The forward error correction technique incorporates three levels of channel coding which interact adaptively. The motivation for using a combination of forward error correction and link-layer retransmissions is to obtain better performance in terms of end-to-end throughput and latency by correcting errors in an unreliable wireless channel in addition to end-to-end correction rather than by correcting errors only by end-to-end retransmissions. The coding overhead is changed adaptively so that bandwidth expansion due to forward error correction is minimized. Integrity of the link during handoffs (in the face of mobility) is handled by window management and state transfer. The protocol has been implemented. Experimental performance results based on the implementation are presented.

1 Introduction

Most data link-layer protocols in existence today have been designed for conventional networks: low-speed landline networks based on voiceband modems which have fixed and relatively low error rates, LANs with very low error rates, the satellite channel with a large propagation delay, or high-speed networks with extremely low error rates and relatively large propagation delays (with respect to the packet size). The mobile wireless channel has different characteristics than these conventional networks: the error rate can be very high and it is highly variable, and the propagation delay with respect to the packet sizes is small. With the increasing popularity of wireless communication, in the form of outdoor cellular communications with small bandwidth, or indoor wireless LAN with large bandwidth, there is a growing need for new link-layer protocols to accommodate the specific properties of the wireless channel in an efficient way so that a wireless terminal with better performance, lower power, and smaller size can be designed (see, e.g., [KGBK78], [LNT87]). In this paper we propose such a protocol. We envision a network architecture which has a wired network as its backbone with base stations on it acting as access points for the mobile terminals (Figure 1). Mobile terminals communicate through the base stations with other hosts. In the context of this architecture, a *wireless channel* refers to the channel between a wireless terminal and a base station. Our emphasis throughout the paper will be on the link between the base station and a mobile terminal.

The configuration with a wireless channel between a base station (which is wired to the backbone network) and a mobile terminal (which has no wired connection) is inherently *asymmetric* in nature. The asymmetry can be attributed to the fact that the mobile terminal has *limited power* and *smaller processing capability* than the base station. In order to accommodate this asymmetry, we propose to put as much *intelligence* as possible in terms of processing in the base stations and make the mobile terminals *relatively* dumb. This issue will be addressed in Section 2.

Because of the effects of fading, interference and mobility, the error rate incurred in a mobile wireless channel is often very high. This has two effects. First, in current systems, end-to-end protocols must recover from these errors, often using retransmission timers. These timers are typically set to values on the order of seconds to allow for variable network delays. This causes the recovery time of an error that occurs on a wireless channel to be long. Secondly, losses incurred on the wireless channel trigger the end-to-end transmitters to decrease the window size and increase the duration of the retransmission timers as observed by Caceres and Iftode in [CI93]. This leads to lower throughput and higher latency. Therefore, correcting errors at the link-layer in addition to end-to-end correction will result in better performance compared to only end-to-end correction. However, there is evidence in the literature that a retransmission in the link-layer does not *always* result in better performance owing to its complex interaction with end-to-end retransmission mechanisms [DCY93].

As stated above, the wireless channel has a variable bit error rate, which can be very high ($10^{-2} - 10^{-3}$). In addition, the wireless channel has fixed bandwidth, but the transmission rate needs as well as the number of potential users are constantly increasing. These two properties lead to conflicting requirements in the design of a protocol. Since the bandwidth is limited, it is necessary to minimize retransmissions. On the other hand, since the error rate can be very high, it may be desirable to make redundant transmissions with the assumption that the probability of a packet being delivered correctly is higher if it is transmitted more than once. This tradeoff, which must be taken into consideration in the design of the protocol parameters, as well as the aforementioned need to design the protocol to balance end-to-end performance and link-by-link error control is the subject of forward error control. This issue will be addressed in Section 3.

In cellular communications, as the mobile travels through different cells, the base station that the mobile communicates with changes. This process is known as a *handoff*. For data communications, there have been attempts to reroute or forward messages at the network layer, e.g., Mobile IP [S94]. Our approach provides transfer of packets during handoffs which might be lost in the case of Mobile IP without a protocol such as ours. These lost packets would then have to be retransmitted at the transport layer leading to increased end-to-end delay. By state transfer and link-layer retransmissions, the retransmissions at the transport layer are avoided. This requires a synchronization of the link-layer state at the new base station with the link-layer state at the old base station. We have designed techniques to deal with handoffs at the link-layer, these will be described in Section 4.

The indoor and the outdoor channels show very different error characteristics due to different fading effects. This means they may require different error correction mechanisms. The difference in channel characteristics and, based on these differences, error correction mechanisms for each environment are explained in Section 5. Some performance results based on an implementation of the protocol are given in Section 6. Summary and conclusions are provided in Section 7. Appendix A provides a brief description of the formal specification of the protocol.

Although some protocols such as SDLC or HDLC have asymmetric modes, they differ from the work described here in that they are based on a master-slave relationship. AIRMAIL avoids this relationship and therefore the associated polling operation, waste of bandwidth, and delay.

The Cellular Digital Packet Data (CDPD) protocol has a data link-layer protocol known as Mobile Data Link Protocol (MDLP), based on HDLC and LAPD [CDPD93]. Although partially based on HDLC, MDLP operates in balanced mode, i.e., there is no asymmetry in the protocol. In addition, forward error correction or precautions taken for handling mobility existent in AIRMAIL do not exist in MDLP.

A link-layer protocol for the IS-95 channel, named the Radio Link Protocol (RLP), was proposed in [K93]. RLP is a simple, “lightweight” protocol. RLP emphasizes end-to-end error control, and therefore is not fully reliable. It leaves part of error control to higher layers, e.g., TCP. It is based on a pure-NAK philosophy, i.e., it is “success-oriented.” The end-to-end, success-oriented approach is valid for clean media such as fiber, but for very noisy environments such as cellular wireless channel, we believe there is merit in correcting errors at the error-prone link. With error correction mechanisms at the link-layer, the heavy error correction overhead present in the physical layer of IS-95 can be avoided.

Recently, a new point-to-point ARQ (Automatic Repeat Request) protocol has been proposed for the radio channel [NED93] which exploits the sequential delivery of packets in a wireless link. However, it assumes circuit-mode data, while we assume packet-mode data. Further, [NED93] does not have any of the unique properties of our protocol, namely, asymmetry, adaptive forward error correction, and mobility management.

We describe the asymmetric features of the basic protocol first. We then describe the forward error correction techniques and show how forward error correction and the retransmission mechanisms of the asymmetric protocol can be combined to obtain reliability. Then, how mobility is taken care of in AIRMAIL is described. These descriptions are followed by simulation and performance measurement results. The basic asymmetric protocol which relies only on ARQ techniques is described next.

2 Asymmetry

The key ideas in incorporating asymmetry affect the ARQ error control and window-based flow control techniques used in AIRMAIL. These are summarized below.

1. Timers are *always* at the base station regardless of whether it is transmitting or receiving. Thus the *intelligence* in terms of maintaining timers, processing complex status messages and most importantly, making decisions *resides* in the base station.

2. The base station receiver sends its status to a mobile transmitter *periodically*, as in the SNR protocol [NRS90]. The justification for using periodic status messages is to reduce the dependence on the error-prone medium. In particular, if the wireless channel is bad and the base station does not receive packets, it can still send status messages, because sending of status messages is triggered by the timeout signal of a local timer and not by the event of receiving a packet. Also, if a status message

gets lost, a subsequent status message will shortly follow.

Note that the period of sending status messages should be optimized so as not to waste much bandwidth for control information while at the same time offsetting the effect of a noisy channel.

3. The mobile receiver *does not* send its status to the base transmitter periodically because of its power constraint. These status messages are event-driven.

4. The mobile receiver *combines* several status messages into one status message to conserve power. That is, the mobile terminal does not send a status message after receiving each packet. Rather it sends a status message after receiving a block (a set of packets). However, there is a trade-off between wasting power and increasing latency.

   Thus a mobile terminal *transmits packets*, but does not send an acknowledgment after receiving each packet. It waits for receiving a whole block before sending out a status message.

The following subsections provide more detail on how repeat requests and window-based flow control are handled in AIRMAIL.

2.1 From Mobile Transmitter (MT) to Base Receiver (BR)

The steps involved in providing repeat requests and flow control from MT to BR are listed below (Figure 2):

1. MT transmits new packets continuously until a maximum transmit buffer size is reached (maximum buffer size is computed using a multiple of the round-trip delay), a retransmission request (status message notifying loss/corruption of a packet) is received, or it has no more data to transmit. Retransmission has priority over transmission of a new packet. When MT has no more to transmit, it sets a bit in the packet header of the last packet. This bit is referred to as the *e-bit* in subsequent sections of the paper.

2. BR transmits status messages *periodically* to MT. This is similar to periodic state exchange in the SNR protocol [NRS90] except that state is not *exchanged* and that the status messages in this protocol do not contain information regarding how much buffer space is available in the receiver as in the SNR protocol. The periodicity is maintained using a status timer which expires after a given interval and is restarted after a status message is sent. The period is reduced when BR receives a packet with the e-bit set. In such a case, BR sends its status immediately without waiting for the Status Timer to

expire. This allows for fast acknowledgments when small amounts of data are being sent and quick response is the main concern.

Since MT does not send its status periodically and it does not have a timer, a timer is needed at BR to detect loss of packets. The concept of using a timer at the receiver was introduced in NETBLT [CLZ87], although in the context of a transport protocol. Also note that BR sends its status periodically to MT, thereby incorporating redundancy in the error-prone medium.

3. MT retransmits the requested packets before transmitting any new packet.

   If retransmissions do not reach BR, MT will not be able to transmit any new packets after a finite period because the transmit buffer space at MT will be exhausted. The transmit buffer size *buf* at MT is *larger* than the *window*[1] to take care of loss of status messages from BR to MT. This improves the performance in terms of throughput and delay.

   Since *buf* is larger than the *window*, MT can transmit new packets longer than usual with the hope that status messages acknowledging previously transmitted packets will arrive a little later than the expected time owing to the loss of first few status messages. This prevents idling by the transmitter. In other words, MT *anticipates* opening of window or freeing of buffer space in near future.

   Buffer space at MT will be exhausted if the *forward* channel (BR to MT) is bad or the *reverse* channel (MT to BR) is bad or both are bad. We argue next that in any case such an anomaly will be reflected at BR in the form of an *unchanged* status.

   Consider first the case in which the reverse channel is bad. That is, the (re)transmitted packets from MT to BR are getting lost. In that case, the status of BR will not change because BR is not receiving anything from MT. If however, the forward channel is good, the status messages (which are identical) from BR will reach MT. In that case, MT will keep retransmitting the packets with the expectation that the channel will improve soon. If the channel does not improve, BR will not receive any retransmitted packet and hence its status will not change. The anomaly will thus be detected at BR.

   Consider next the case in which the forward channel is bad. That is, the status messages from BR to MT are getting lost. If the status messages are lost, MT will have no idea as to which packets are received at BR. Thus, MT will soon run out of buffer space and stop transmitting new packets. Not only will MT stop transmitting new packets but it will also not retransmit any old packet before

[1]Throughout this paper the term $window$ is used to represent the number of packets that can be transmitted during a round-trip delay.

it receives a status message from BR. The reason for MT's not retransmitting any old packet is that it does not know which packets have been received by BR and it does not want to retransmit indiscriminately because bandwidth is expensive. Since MT stops any (re)transmission, BR's status will stop changing and the anomaly will be detected.

Thus whether the forward channel or the reverse channel goes bad, the anomaly will be reflected at BR as a no change of its status. BR will drop the connection if such a situation persists for some time.

2.2 From Base Transmitter (BT) to Mobile Receiver (MR)

The steps involved in repeat requests and flow control from BT to MR are listed as follows (see Figure 3):

1. BT transmits new packets until the window closes, a request for retransmission (which is a part of the status message) arrives, or it has no more packets to transmit. When BT has no more packets to transmit, it sets the e-bit in the packet header of the last packet.

   BT starts a timer after transmitting a full block of packets or after transmitting a packet with the e-bit set, and sends an explicit status request message (Poll) to MR if it does not hear from MR before the timer expires. Note the presence of timer at BT.

2. MR sends a status message after it receives a whole *block* (fixed number of packets). The status message is a bitmap such that bit_i indicates whether $packet_i$ within the given block has been received or not. Bit_i = 1 indicates "received" while bit_i = 0 indicates "not received."

   Note that to conserve power, MR, unlike BR (when Base Station is in the receiving mode), *does not* send status messages periodically (at fixed intervals of time) and frequently (more than once during the round trip time). Moreover, for the same reason, it does not send a status message after it receives each packet. MR waits for either a whole block, or a packet with the e-bit set, to arrive before sending out a status message. This allows the mobile unit to conserve transmission power.

3. BT retransmits the requested packets.

   Note that BT selectively retransmits only the "lost" packets in a given block and not the whole block.[2] This conserves bandwidth.

[2]This is different from the SNR protocol in which a whole block is retransmitted. SNR protocol is designed for high-speed networks where bandwidth is not expensive and hence retransmitting a whole block is not a big issue. However, in wireless links, bandwidth is expensive and hence we minimize all transmissions.

BT also starts the timer to detect if MR sends a status message acknowledging the receipt of the retransmitted packet within the expected time. If not, BT sends an explicit Poll message to MR. If the status message from MR does not arrive before the timer expires, BT tries again and gives up after a few attempts if MR does not respond.

The basic asymmetric protocol relies only on the retransmission mechanism to ensure reliability. Deciding whether errors are present in a packet is carried out by means of a cyclic redundancy check. By increasing the power of the code used for that purpose, it is possible to correct errors and avoid retransmissions. In addition, errored packets that cannot be corrected by means of this redundancy can be corrected by additional parity packets. For real-time traffic, such as audio and video, retranmissions are not feasible, and forward error correction is viable for ensuring a given quality of service. By changing the parameters of forward error correction, the best channel utilization can be achieved. These ideas are described in more detail in the following section.

3 Forward Error Correction

The digital mobile channel has various sources of random noise and deterministic interference that give rise to errored receptions. The most dominant factors are Rayleigh fading due to Doppler shift of the operating frequency for mobile transceivers or mobile scatterers, multipath interference due to reflections from natural and man-made objects around the main transmission path, random noise, and co-channel and adjacent channel interference. Various signal processing alternatives are under consideration for these sources of interference, such as equalization, space diversity, multitone transmission, and forward error correction (FEC) at the physical layer. No consensus exists in the literature as to which of these techniques can universally recover from all sources of noise and interference. It has been generally accepted that some FEC at the physical layer is needed to recover from noise for the outdoor channel. In general, however, this FEC will not be able to generate an error-free channel, and some residual error will be propagated to the data link layer.

As is well-known, FEC techniques are sometimes used in addition to the automatic repeat request (ARQ) techniques at the data link layer. Various combined techniques, known as hybrid techniques, have been suggested and studied for this purpose (e.g., see [LC83] and its references). Among the channels suggested for the use of hybrid ARQ/FEC schemes is the satellite channel and the high-speed ATM channel. In both cases, the packet size is large with respect to the propagation delay, or in other words, the bandwidth-delay product is high. This makes the process of requesting and receiving retransmissions take long with respect to the packet size, and consequently reduces the protocol throughput. Therefore FEC, by means of adding redundant packets so that errored or “lost” packets can be reconstructed based on the redundant packets and packets that are correctly received, becomes a feasible alternative to ARQ. The land-mobile channel does not have a large delay-bandwidth product, and therefore, the use of FEC at the data link-layer as suggested above may appear to be questionable. We will address this issue in this section, and will summarize under which circumstances FEC becomes useful. First, we would like to discuss possible ways of implementing FEC in a data transmission system.

There are three possible levels FEC can be incorporated into a data transmission system.

- Bit-level FEC: This is achieved at the physical layer, typically in hardware, by means of a DSP chip or an application specific integrated circuit. For bandwidth-limited channels, trellis coded modulation with Viterbi decoding is used. If the channel is not bandlimited, block or convolutional coding techniques are employed. In the latter case, decoding is by means of the Viterbi algorithm. A characteristic of the Viterbi algorithm is that it provides a sequence, which is closest to the original given the received sequence in some mathematical measure (i.e., maximum likelihood). By definition, the Viterbi algorithm does not provide any indication of an uncorrectable sequence, or the number of corrections, etc. Although it is possible to correlate calculated values in the algorithm with the long term channel bit error rate, the Viterbi algorithm does not provide a measurement of the short-term channel error characteristics based on channel errors.

- Byte-level FEC: This is done by means of per-packet FEC. Every packet in a data link-layer protocol carries a cyclic redundancy check (CRC) field in order to determine whether the packet is received error-free. The same field can be used for error correction. Most traditional data link-layer protocols use this field for error detection only, mainly due to the computational complexity of performing FEC decoding. Recently, with the advent of more powerful processing, additional use of this field for error correction purposes is being considered. For example, ATM AAL 3/4 has a per-cell CRC field which is designed to be capable of one bit error correction. In a similar fashion, we propose replacing this field by a code that is capable of error correction, such as the output of a Reed-Solomon encoder. When the field size is the same as that needed for CRC, there is no additional bandwidth or coding overhead required as compared to an ARQ scheme, although there is an associated increase in the probability of undetected errors. By increasing the size of this field, the probability of undetected error and the number of packet retransmissions needed can be reduced. Our simulations indicate most blocks have a few bytes in error for the outdoor channel, and therefore, bit- and byte-level FEC can be highly beneficial for this environment. Byte-level FEC has the advantage of correcting only those bytes in error (as the error process manifests itself as short bursts), and therefore has the advantage of using the bandwidth more efficiently. The interplay between these two levels of FEC is an interesting area of research.

- Packet-level FEC: This is done by allocating some packets in the protocol window for correction. In the case of packets that cannot be corrected by bit-level or byte-level FEC (or equivalently, packets that are "lost"), these redundant packets, together with the correctly received packets, can be used for recovery of the lost packets without retransmissions. It is possible to add M redundant packets to N data packets, so that as long as at least some N packets of the total $N + M$ data packets and redundant packets are received, the N data packets can be recovered. This technique is known as optimum erasure encoding-decoding (optimum in the maximum distance sense), and various coding methods can be employed for this purpose. A well-known technique is Reed-Solomon codes, which can be used for error correction (positions unknown, as in byte-level FEC) or erasure correction (positions known) [LC83]. When the number of packets is 2 or 3, the diversity code [AIGM93] is more efficient in terms of the size of the field the operations are performed in. Again, with the advent of more complicated processing, FEC methods at the packet level are finding their way into data transmission. For example, there is a proposal to incorporate cell-level FEC into ATM AAL 1.

In the system we are proposing, all three levels of FEC have their place. Our simulations have shown that different mobile channels show different characteristics. The narrowband outdoor mobile channel has a small number of bytes in error for most of the packets transmitted. There exists some correlation in the number of errored bytes with time. This correlation can be exploited for estimation of the number of bytes to be in error for the next block, and the FEC overhead can be changed by a message from the receiver to the transmitter accordingly. This is equivalent to measuring the channel continuously and describing the channel "state" to the transmitter. It has a similar motivation to the periodic state exchange of the SNR protocol [NRS90]. The rate of these measurement messages is orders of magnitude smaller than the rate of power control messages (1 kb/s) for IS-95 digital cellular standard using CDMA. The small delay-bandwidth product of the mobile channel enables adaptation based on feedback. Our simulations have shown better channel utilization by using an adaptive algorithm as described above. The messages from the receiver are in general used to change the FEC parameters at all levels. For the narrowband outdoor channel, bit-level and byte-level adaptation is dominant. Changing of the physical layer parameters from measurements by the data link-layer is accomplished by sending messages to the bit layer. For trellis coded modulation or convolutional coding at the data link-layer, the measurement of the channel error characteristics cannot be obtained as a byproduct of the decoding process at the physical layer and therefore these messages are essential for adaptation at the bit level. The bit-level and the byte-level adaptive FEC are well-suited to the narrowband outdoor mobile channel.

On the other hand, the wideband indoor channel shows very different error characteristics. This channel is error-free most of the time. As the RMS delay spread to bit period ratio approaches 10%, however, the channel begins to show error characteristics such that most packets within a window are error-free, but some have very large number of bytes in error. These cannot typically be corrected by bit-level or byte-level FEC, and use of packet-level FEC is therefore in order for this application. The wideband outdoor channel will benefit from all three levels of coding.

As in ATM, exclusive use of FEC is needed for real-time applications, such as voice or video. Data, on the other hand, can be transmitted by a combination of ARQ- and FEC-based data transmission protocol.

The issue of using packet-level FEC with a sliding window ARQ algorithm requires some bookkeeping. In order to illustrate how this can be done, we give a simple example. Consider a base transmitter and a mobile receiver. Suppose the window size is $W = 8$ and $M = 2$. The transmitter transmits $N = 6$ data packets (say $d0$, $d1$, $d2$, $d3$, $d4$ and $d5$) followed by $M = 2$ parity packets (say $p0$ and $p1$ computed based on the data packets $d0$, $d1$, $d2$, $d3$, $d4$ and $d5$). If the receiver receives *any* 6 of the 8 packets (say packets $d0$, $d2$, $d3$, $d5$, $p0$ and $p1$), it can reconstruct the lost data packets $d1$ and $d4$. However, if it receives less than 6 packets, recovery is not possible.

For example, suppose the receiver receives only 4 packets $d0$, $d1$, $d4$ and $p1$. In that case, it cannot reconstruct data packets $d2$, $d3$ and $d5$ and hence sends a retransmission request to the transmitter in the form (L_r = 2, Bitmap = 0010). L_r sets the lower end of the window (leftmost bit of the bitmap corresponds to the status of data packet L_r) and hence the Bitmap 0010 corresponds to the status of data packets d2,d3,d4 and d5 respectively from left to right. The transmitter moves L_t (lower end of the window at the transmitter) to 2, retransmits data packets $d2$, $d3$ and $d5$, transmits *new* data packets $d6$ and $d7$ and computes 2 parity packets $p0'$ and $p1'$ based on $d2$, $d3$, $d4$, $d5$, $d6$ and $d7$ and transmits them following the data packets. Thus the parity packets are computed based on the data packets in the *current* window. The information about which packets have been used in computing the parity packets is available at the receiver (because it is the receiver which sends the bitmap and L_r) and hence, the receiver can use that information to recover lost data packets from the packets it receives at any instant of time.

A combination of FEC and the retransmission mechanisms of the asymmetric protocol is sufficient to ensure reliability of a wireless link as long as the two endpoints of the link, in particular, the base station, remains the same. When the mobile terminal moves from one cell to another, and therefore changes base station, packets may be lost during transition, resulting in inconsistent states at the mobile terminal and the new base station. In order to be able to maintain a reliable link in the face of mobility, we take additional measures at the link-layer which are described in the next section.

4 Mobility and Handoff

Handoffs occur when the communication to and from a mobile terminal is transferred from one base station to another. To allow this link-layer protocol to operate in a mobile environment, provisions must be made for handoffs to occur. For example, when a handoff occurs, the sequence numbers of the link-layer protocol in the mobile terminal and the new base station must be synchronized so that lost packets may be detected, and ordered data delivery may be provided to the higher layer protocol. In contrast to [KMSKF93], where only real-time services are addressed, we aim for reliable data transfer; and in order to limit the amount of extra processing that must occur at the mobile terminal, we infuse the new base station with the state of the old base station so that no change is noticed at the link-layer of the mobile terminal. To enhance the performance of the handoff mechanism, we allow the mobile transmitter to continue transmitting data while the handoff is occurring.

The basic scheme works as follows (see Figure 4):

1. The mobile terminal issues a handoff request to a new base station to which it wishes to handoff. This request includes the state information of the mobile transmitter and receiver.

2. The new base station requests state information from the old base station. This information is used to infuse the new base station with the proper state.

3. While waiting for the information from the old base station, the new base station attempts to derive its proper state from the information provided by the mobile terminal. During this time, it sends no status messages to the mobile transmitter.

4. The new base station buffers data arriving from the network and the mobile terminal. The transmit window of the mobile transmitter is increased during handoff (dynamic windowing).

5. The new base station receives the state information from the old base station, updates its state, and processes all buffered data according to the normal protocol rules. At this time, it generates a status message to the mobile transmitter and operates normally.

Recall from Section 3 that there are three levels of FEC. In order to decide which levels of FEC need to be used in AIRMAIL, we set up a simulation environment for indoor and outdoor channels. The simulation environment and the results are described next.

5 Wireless Channel Simulation Results

In this section we will describe the wireless channel simulations that result in the design decisions of Section 3 briefly. With the purpose of studying the characteristics of indoor and outdoor channels for mobile communications, a simulation setup was programmed (see Figure 5). It is well-known that the mobile channel has various effects that limit the quality of data transmission. The most important of these is due to multipath fading, i.e., various natural and man-made objects reflect or scatter carriers such that the receiver receives the transmitted signal over multiple paths. In each such path, there are random amplitude and phase fluctuations. Furthermore, the mobile causes Doppler shift for each such path. The combined effect of these events is that the magnitude of the received signal becomes Rayleigh distributed, which can become vanishingly small. The Rayleigh distribution is the cause of the name Rayleigh fading for this event. Such channels are closely approximated by Jake's model [J74]. Such a model was programmed and tested for various indoor and outdoor applications using differential quadrature phase shift keying for a carrier frequency of 850 MHz, for data rates 64 kb/s and 2 Mb/s, with blocks of size 256 bytes, mobile speeds 5, 35, and 55 mph, and delay spread (RMS delay of various paths) values of 100 ns (indoor channel) and 3 μs (outdoor channel). Typical outdoor channel results are shown in Figure 6 for a 64 kb/s channel, vehicle speed 35 mph, and delay spread 3 μs. As can be seen, most blocks have errored bytes and the number of errored bytes vary significantly, but there is a correlation in short term behavior of the number of bytes in error that can be exploited for changing the bandwidth adaptively. As the number of errored bytes within a block is usually small, byte-level forward error correction is the appropriate technique to use. On the other hand, typical indoor channel results are shown in Figure 7 for a 2 Mb/s channel, in an indoor environment when people are walking in the building (delay spread is 100 ns, speed 5 mph). In this case the channel is usually error-free, with a large number of bytes being in error occasionally. When a block has errors, their number is high so that byte-level forward error correction will not work. On the other hand, the periodicity of errored blocks indicates packet-level FEC will work under these circumstances. Results shown in Figure 7 and 6 indicate that bit- and byte-level FEC is appropriate for the narrowband outdoor channel, while packet-level FEC is appropriate for the wideband indoor channel.

The protocol was specified using Communicating Extended Finite State Machines (CEFSMs), and using this specification, was translated into textual APSL (Augmented Protocol Specification Language). Part of the formal specification of the protocol is given in Appendix A. The APSL validator was then used to explore the state space of the protocol in a random manner. The reason for random exploration is to avoid the well-known state space explosion. After validation, the protocol was implemented in several steps. The first implementation of the asymmetry properties was carried out in the user space with Unix sockets. At this point, a SunOS 4.1.3 kernel implementation exists as a loadable kernel module. We made several measurements based on the implementation, which are described next.

6 Experimental Performance Results

The objective of this section is to justify the claims we made about the key properties of the protocol, namely, conserving bandwidth by preventing redundant retransmissions and conserving power by combining several status messages into one by the mobile receiver even in the absence of timers. We also substantiate our claim of asymmetry by the size of code and the processing time at the base station and at the mobile terminal. These results are obtained from the socket-based implementation of the protocol.

First of all, we show that the protocol minimizes redundant retransmissions. This conserves the expensive cellular bandwidth. Figure 8 shows 3 separate plots corresponding to different packet error rates. The number of status messages on the horizontal axis is varied by changing the timeout interval of the status timer at the base receiver. The plot shows that the number of packets retransmitted by the mobile transmitter is the same irrespective of the number of retransmission requests (the slight variations correspond to different number of packets lost in different runs), and that number is *exactly* equal to the number of lost packets. That is, if we plot the number of lost packets on the same axes, it would overlap completely with the plot for number of retransmitted packets. Thus, even if the base receiver sends its status messages very frequently, the mobile transmitter prevents duplicate retransmission. The same is true for the base transmitter.

The next plot (Figure 9) shows that there is a trade-off between sending the status more frequently to obtain a better response time, versus the wastage of expensive bandwidth. In this case, the timeout interval of the status timer at the base receiver is altered to send status messages at different frequency to the mobile transmitter. If the status messages are sent too frequently, bandwidth is wasted. However, the response time improves. If the status messages are sent infrequently, bandwidth is preserved but response time suffers. However, the good news is that there is an optimal choice of the timeout interval for the status timer such that the gain in terms of response time is maximum at the expense of minimum additional bandwidth. From Figure 9, the optimal choice is to send 1.5 to 2 status messages per round-trip delay to the mobile transmitter.

Our measurements indicated that, as expected, the throughput falls with higher packet error rates. However, the normalized throughput of the protocol does not fall below 0.8 even for a packet error rate of 0.1.

We show the effects of asymmetry in the protocol design by providing some figures from the socket-based implementation. Table 1 compares the size of compiled code at the base station and at the mobile terminal. It shows that the total size of code at the mobile terminal is two-thirds the size of the code at the base station.

Table 2 shows the average processing time for the mobile transmitter and the base receiver for transferring a file of size 204 KBytes at a packet error rate of 0.33 and status timer timeout interval equal to half the round trip delay. It shows that the average processing time for the mobile transmitter is one-third of that of the base receiver. The difference between the two reduces with decrease in error rate.

Table 3 shows the average processing time for the base transmitter and the mobile receiver for transferring a file of size 204 KBytes at a packet error rate of 0.01 and poll timer timeout interval equal to twice the round trip delay. It shows that the average processing time for the mobile receiver is two-thirds of that of the base transmitter. The difference between the two increases with increase in error rate.

7 Conclusion

This paper presents the design and performance of a novel link-layer protocol for a digital cellular channel. The protocol is asymmetric in the sense that the intelligence in terms of making decisions is always at the base station, irrespective of whether it is transmitting or receiving. The mobile terminal follows a relatively simpler protocol compared to the base station, mainly because it has limited battery power and smaller processing capability. We showed how the protocol conserves bandwidth by preventing redundant retransmission and how the mobile terminal conserves power by combining several status messages into a single comprehensive status message. The use of adaptive forward error correction with retransmission is particularly suitable for error-prone wireless channels. The forward error correction technique encompasses bit-, byte-, and packet-level channel coding. The interplay between these three levels of channel coding is managed by means of an adaptive algorithm. We showed how forward error correction can be added to our basic protocol. We described a link-layer approach to mobility management based on window management and state transfer.

The protocol described here has been implemented for the Unix kernel. Its unique properties make it well-suited for the digital cellular channel. This paper is an overview of the main ideas in the protocol.

Upcoming papers will describe the protocol in detail, in terms of a formal specification with state diagrams, detailed description of the adaptive forward error correction algorithm and mobility management, and will provide performance analysis and comparisons.

8 Conclusion

Thanks are due to Shilpa Talwar for her help in running simulations for characterizing the mobile channel.

Appendix

A Formal Specification of the Protocol

In order to describe the protocols in a formal way, we represent them in the form of Communicating Extended Finite State Machines (CEFSMs). In order for an EFSM to transition from one state to another, it may require certain *conditions* to be true (which depend on the values of the context variables), or may require certain *inputs* to arrive (from another communicating machine). In addition to that, these EFSM's may *update* context variables or send certain *outputs* (to other communicating machines) during a transition from one state to another. The *condition*, *input*, *output* and *update* are represented as follows:

1. A condition is represented by $\{c\}$.

2. A message is represented by M where M can be of the form $m_1!x$ (send a message x to EFSM m_1) or $m_1?x$ (receive a message x from EFSM m_1). The message x, in general, can have several parameters.

   An abbreviation ($m_1?x + m_2?y$) will be used to represent either $m_1?x$ or $m_2?y$. Similarly, the abbreviation ($m_1!x * m_2!y$) will be used to represent $m_1!x$ followed by $m_2!y$.

3. An update is represented by $[u]$ where u can be thought of as a simple program segment which updates context variables. The update may depend on conditions. Thus, u may have conditional statements as well as assignment statements.

We only give the formal description of the protocol between MT and BR for brevity.

The mobile transmitter MT is a single EFSM (Figure 10). MT maintains an array T_array in which each entry is a record with three fields: abit, clock and ebit. An entry $T\_array[i].abit$ is 1 or 0 depending on whether packet i has been acknowledged or not. Similarly, an entry $T\_array[i].clock$ contains the local clock time when packet i was last (re)transmitted. $T\_array[i].ebit$ = 1 or 0 depending on whether packet i indicates an early end of a block or not. MT maintains a context variable *nop* (number of outstanding packets) which is the difference between the highest numbered packet transmitted (H_t) and the lowest numbered packet acknowledged (L_t). *Maxbuf* is a constant representing the maximum size of buffer at MT. We also assume that MT sends an indication when it has no more packets to send. The indication which depends on the implementation may be in the form of a bit or in the form of a special packet. Here we assume that the indication is in the form of a bit which we call *ebit*. The variable *ebit* is also set to 1 in a packet if that is the last packet in a window. Otherwise it is 0.

The EFSM of the base receiver BR communicates with the Status Timer(ST), which is also modeled using an EFSM (Figure 11). We assume that BR maintains an array R_array such that an entry $R\_array[i]$ is 1 or 0 depending on whether packet i has been received or not. BR maintains two pointers L_r and H_r. L_r points to the lower end of the window. That is, all packets until L_r - 1 have been received and L_r is the first packet which has not been received. H_r points to the higher end of the window indicating the last packet which has been received. Thus if L_r = H_r + 1, then all packets have been received so far. BR sends its status when ST expires. BR sends $stat\_ack(L_r)$ if L_r = H_r + 1 (that is, it has received all packets so far). BR sends $stat\_ret(L_r, w, Bitmap)$ if there is an entry $R\_array[i]$ = 0 for some i between L_r and L_r + w, where w = H_r - L_r + 1. BR maintains a context variable *nochng* which represents the number of times the status of BR remains unchanged and is incremented if the status of BR does not change since the last time it is sent. The constant *MaxNochng* indicates the maximum number of times the status of BR is allowed to remain the same before disconnection.

Finally, we give a formal description of handoff between a mobile transmitter and a base receiver and show how it fits into the formal specification of the protocol without handoff (Figure 12).

A base station receiver is initially in state idle. When it receives a handoff request (ho_rqst) from a mobile transmitter, it transfers into ck_state. The handoff request includes the identity of the old base station and the values for the H_t and L_t variables at the mobile transmitter. The new base station receiver generates a handoff indication (ho_ind) to the old base station. This indication requests that the old base station forward its state information to the new base station.

If L_t is equal to H_t, there is no data outstanding between the mobile transmitter and the new base station receiver. The new base station sets its variables and enters the wait_info state. While in this state, any data that arrives from the mobile transmitter is buffered by the base station receiver (buffer). When it receives the state information from the old base station, the new base station updates its context variables, generates an acknowledgment to the old base station, generates a status message to the mobile transmitter, and moves to the Active state. In the Active state, the base station receiver processes all of the data buffered during the handoff process, and any new data that arrives, according to the normal protocol rules.

If L_t is less than H_t, data is outstanding between the new base station receiver and mobile transmitter and the base station receiver enters state wait_missing. In this state, the new base station receiver buffers any data it receives (buffer_missing) until it receives the state information from the old base station. At this time, it processes the *R_array* received from the old base station. The new base station requests (get_data(i)) that the old base station forward it any data (data(i)) that it has received and buffered. When this has been completed, the new base station contains all data that has been previously transmitted by the mobile terminal that has been either not forwarded to the higher layer protocol by the old base station or lost in transmission. The new base station receiver generates an acknowledgment to the old base station, a status message to the mobile transmitter, and transitions into Active state. While in the Active state, the new base station processes any buffered data and newly arriving data according to the normal protocol rules.

When a base station in Active state receives a handoff indication from a new base station, it sends the new base station its state information. If the old base station receives any requests for data it has buffered (get_data(i)), it sends this data to the new base station (data(i)). When the old base station receives an acknowledgment from the new base station it clears its record and enters the idle state.

During a handoff, it is possible that the window of the mobile transmitter will close. This is because the new base station does not generate any status messages until it has completely synchronized its state. To solve this problem we use a technique called dynamic windowing.

When the mobile transmitter requests a handoff, it increases its window size temporarily. This allows it to continue transmitting past its normal window size. The transmit window is reduced to its proper size when the status message is received from the new base station.

References

[AIGM93] E. Ayanoglu, C.-L. I, R. D. Gitlin, and J. E. Mazo, “Diversity Coding for Self-Healing and Fault-Tolerant Communication Networks,” *IEEE Transactions on Communications,* Vol. COM-41, pp. 1677–1686, November 1993.

[CDPD93] *Cellular Digital Packet Data System Specification Release 1.0,* Book III, CDPD Industry Input Coordinator, Costa Mesa, CA, July 1993.

[CI93] R. Caceres, and L. Iftode, “The Effects of Mobility on Reliable Transport Protocols,” *Proc. of the 14th ICDCS,* June 1994.

[CLZ87] D. D. Clark, M. L. Lambert, and L. Zhang, "NETBLT: A bulk data transfer protocol,” *Proceedings ACM SIGCOMM ’87 Symposium,* pp. 353–359, Stowe, Vermont, August 1987.

[DCY93] A. DeSimone, M. C. Chuah, and O. C. Yue, “Throughput Performance of Transport Layer Protocols over Wireless LANs,” *Proceedings of IEEE GLOBECOM ’93,* pp. 542–549, Houston, November 1993.

[J74] W. C. Jakes (Editor), *Microwave Mobile Communications,* Wiley, 1974 (Reissued by IEEE Press, 1994).

[K93] P. Karn, “The Qualcomm CDMA Digital Cellular System,” *USENIX Mobile and Location-Independent Symposium Proceedings,* pp. 35–40, Cambridge, Massachusetts, August 1993.

[KGBK78] R. E. Kahn, S. A. Gronemeyer, J. Burchfiel, and R. C. Kunzelman, “Advances in Packet Radio Technology,” *Proceedings of the IEEE,* Vol. 66, pp. 1468–1496, November 1978.

[KMSKF93] K. Keeton, B. A. Mah, S. Seshan, R. H. Katz, and D. Ferrari, “Providing Connection-Oriented Network Services to Mobile Hosts,” *USENIX Mobile and Location-Independent Symposium Proceedings,* pp. 83–102, Cambridge, Massachusetts, August 1993.

[LC83] S. Lin and D. J. Costello, *Error Control Coding: Fundamentals and Applications,* Prentice-Hall, Englewood-Cliffs, NJ, 1983.

[LNT87] B. M. Leiner, D. L. Nielson, and F. A. Tobagi, “Issues in Packet Radio Design,” *Proceedings of the IEEE,* Vol. 75, pp. 6–20, January 1987.

[NED93] S. Nanda, R. Ejzak, and B.T. Doshi, “A Retransmission Scheme for Circuit-Mode Data on Wireless Links,” accepted for publication in *IEEE Journal on Selected Areas in Communications.*

[NRS90] A. N. Netravali, W. D. Roome, and K. K. Sabnani, "Design and Implementation of a High Speed Transport Protocol," *IEEE Transactions on Communications,* Vol. COM-38, pp. 721–730, November 1990.

[S94] W. Simpson, “IP Mobility Support,” Submission to Internet Engineering Task Force, September 1994.

	Size of Object File (Bytes)
MT	40,960
BR + ST	65,536
BT + PT	81,920
MR	57,344
Total at Base	147,656
Total at Mobile	98,304

Table 1: Comparison of the size of compiled code.

Window Size	MT Processing Time (seconds)	BR Processing Time (seconds)
4	0.07	0.23
8	0.07	0.23
16	0.07	0.21
128	0.06	0.20

Table 2: Comparison of the processing time for MT and BR.

Block Size	BT Processing Time (seconds)	MR Processing Time (seconds)
4	0.23	0.14
8	0.24	0.16
16	0.26	0.16
32	0.36	0.23

Table 3: Comparison of the processing time for BT and MR.

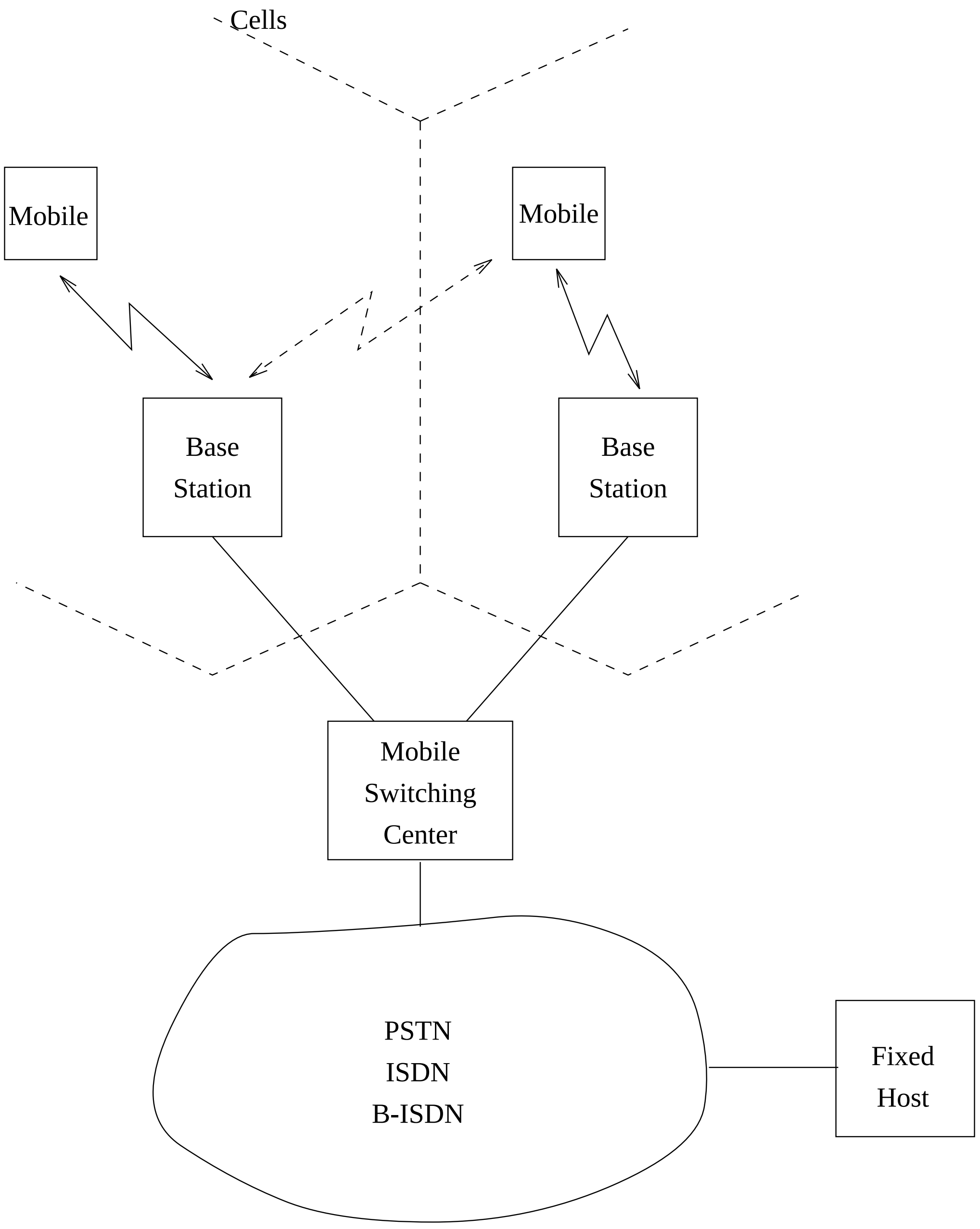


Figure 1: Cellular wireless network architecture. The mobiles communicate via the base stations and the mobile switching center with the backbone network. The backbone network can be the Public Switched Telephone Network (PSTN), or Integrated Services Digital Network (ISDN), or the Broadband-ISDN (B-ISDN). Each base station communicates with the mobiles within the cell that it controls. When mobiles move from one cell to another, the base station that a mobile communicates with changes, which is known as a handoff. A similar scenario is applicable for the LAN environment where the backbone network is a LAN or a centralized switch, and the function of the mobile switching center is integrated into the base stations or the LAN.

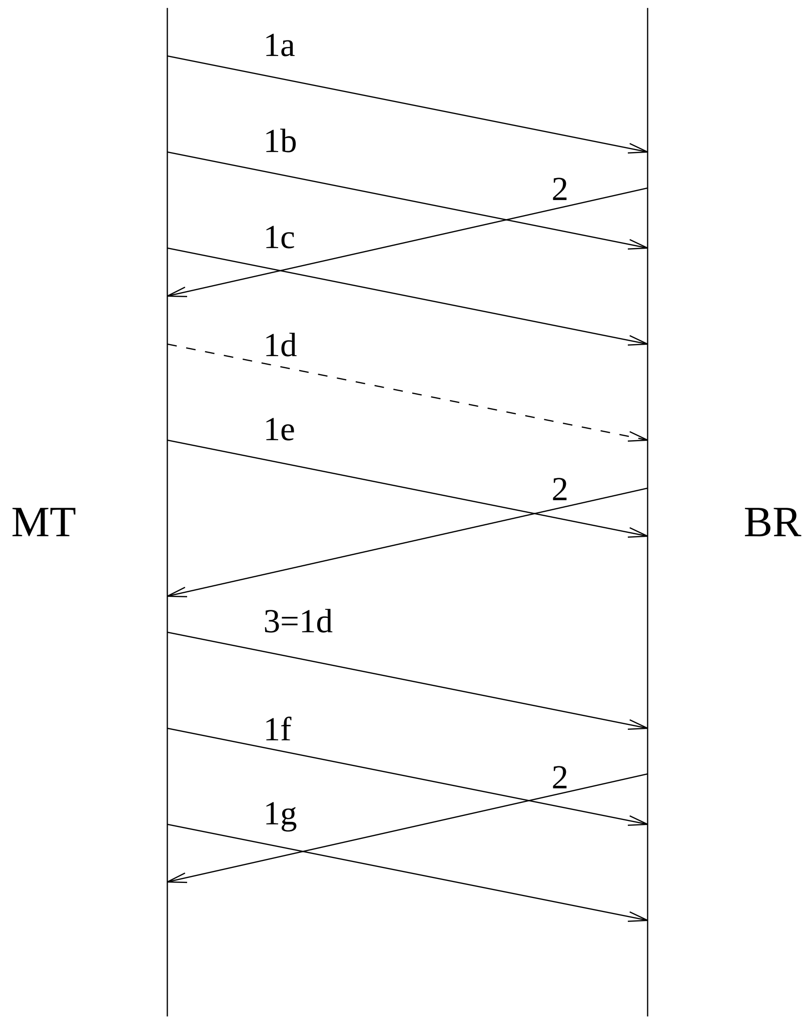


Figure 2: Data transfer from Mobile Transmitter (MT) to a Base Receiver (BR). Messages 1 are new packets transmitted by MT. Messages 2 are periodic status messages transmitted by BR. Messages 3 are retransmissions of lost packets. In the figure, message 1d is lost, and retransmitted upon receipt of the status message.

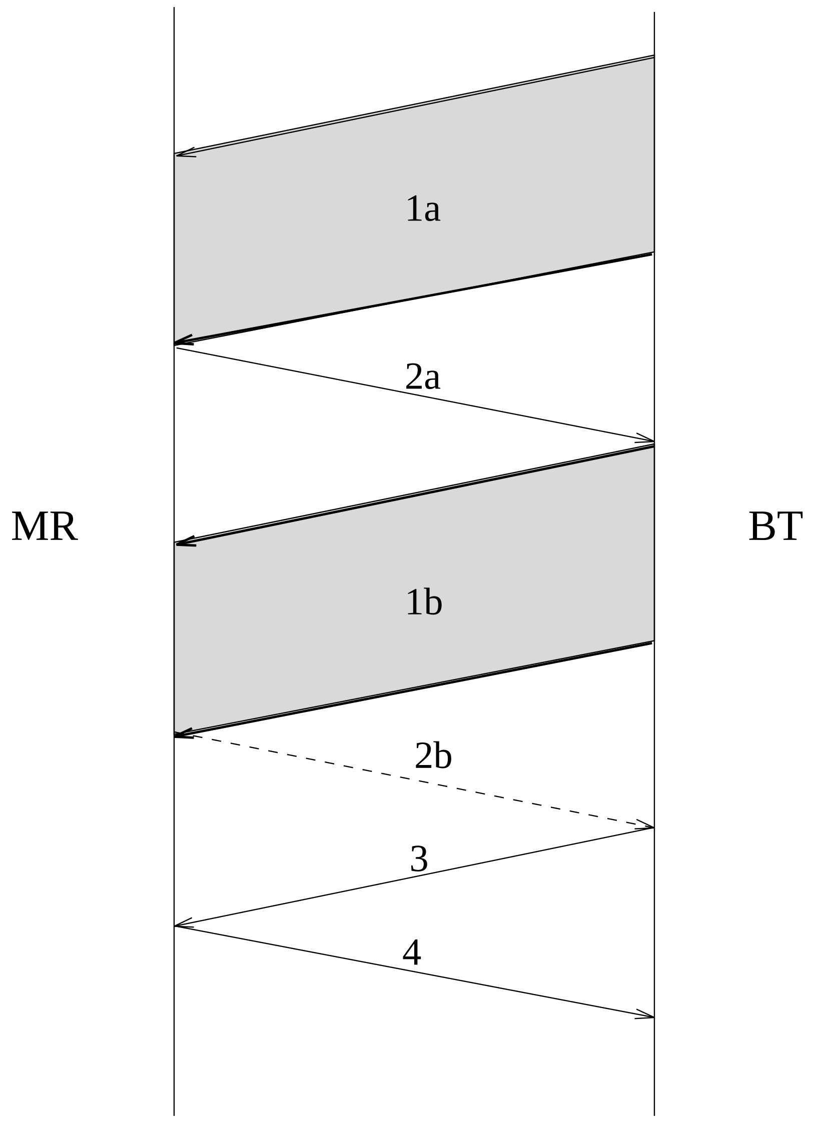


Figure 3: Data transfer from a Base Transmitter (BT) to a Mobile Receiver (MR). When BT completes the transmission of a block (1), it starts a timer. MR transmits a bitmap of indicating which packets within a block have been received (2) when it receives a block. IF BT does not receive the bitmap, it sends explicit status message request (3). When MR receives message 3, it transmits status message (4). In this figure, there is no loss in the block 1a, but there are losses in block 1b. In addition, block acknowledgement 2b is lost. Then, BT sends explicit status message request 3, and status message 4 is transmitted.

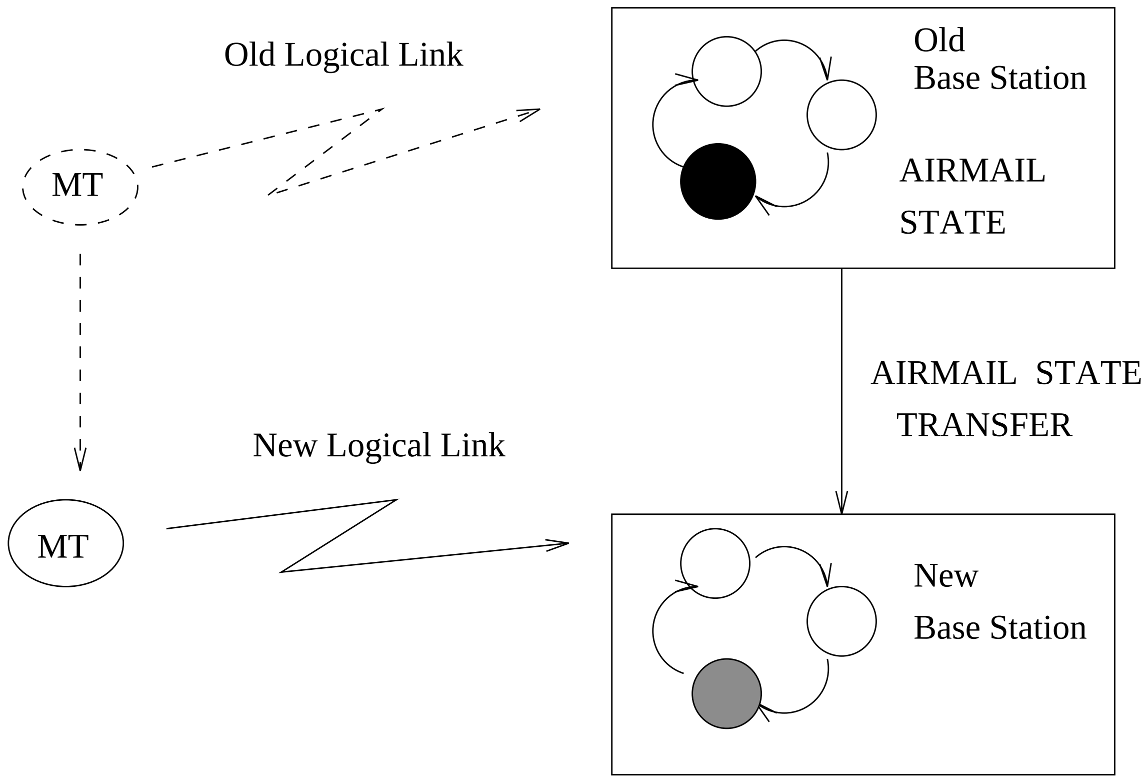


Figure 4: Mobility management in AIRMAIL.

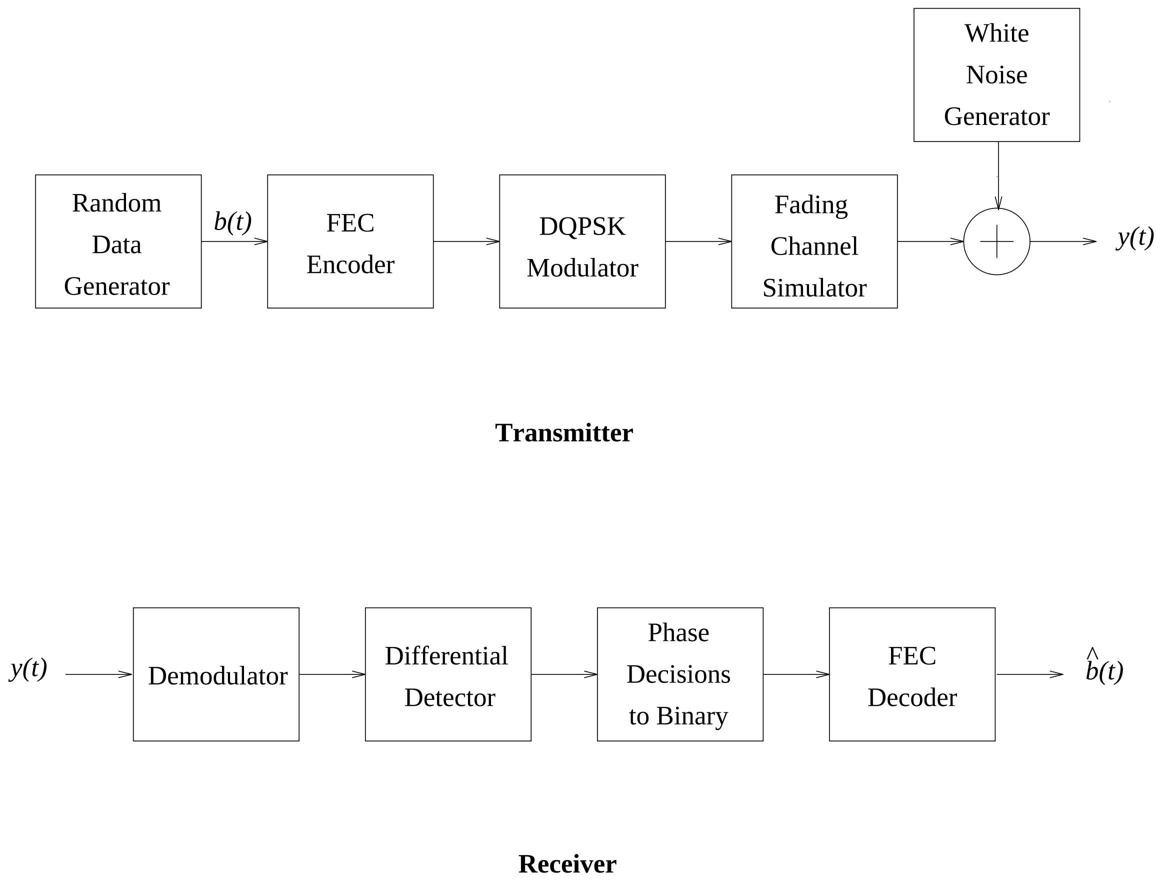
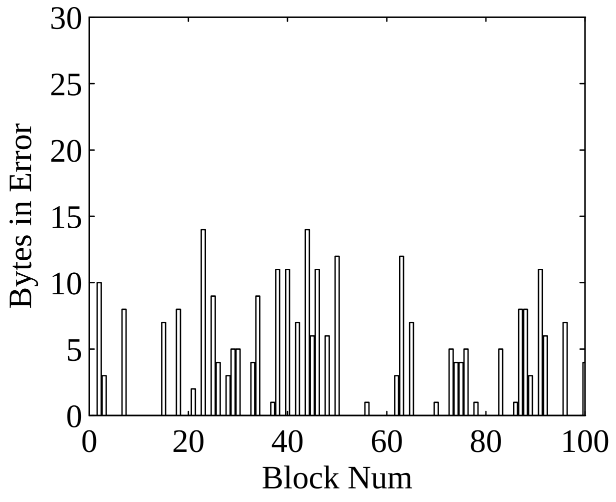


Figure 5: Simulation setup. The input to the FEC encoder at the transmitter, $b(t)$, are bits. The output $\hat{b}(t)$ from the FEC decoder at the receiver are estimates of $b(t)$.



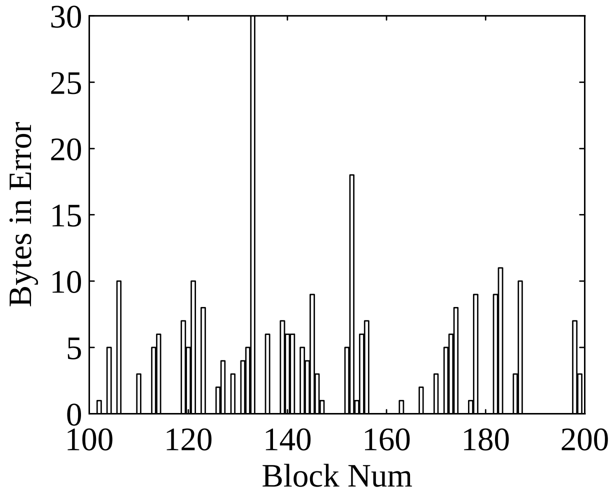
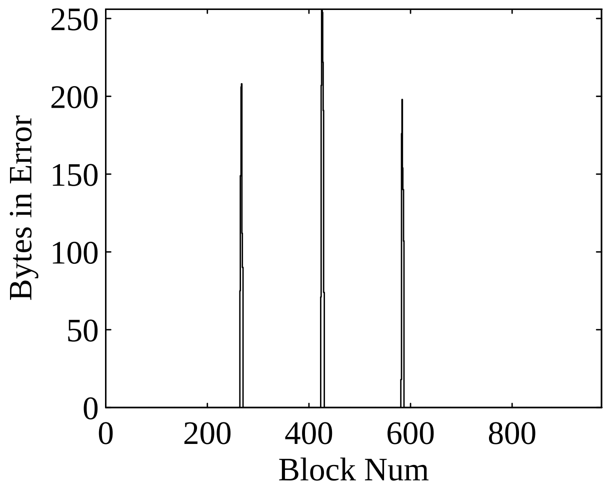


Figure 6: Simulated outdoor channel error characteristics. Number of bytes in error for each block transmitted is plotted against the block number. Each block consists of 256 bytes. Transmission rate $=$ 64 kb/s, vehicle speed $=$ 35 mph, delay spread $=$ 3 μs.



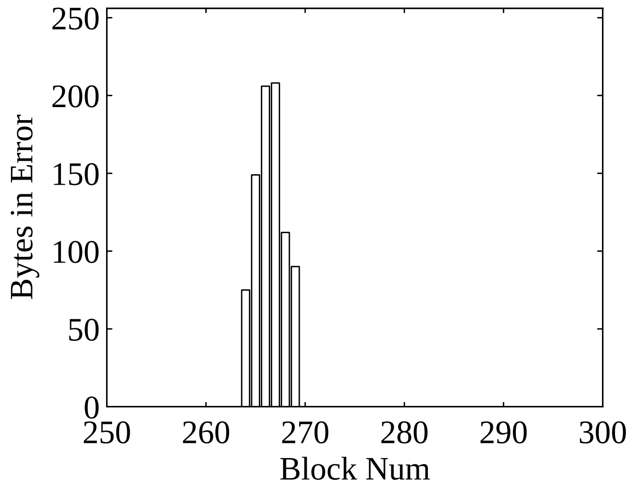


Figure 7: Simulated indoor channel error characteristics. Number of bytes in error for each block transmitted is plotted against the block number. Each block consists of 256 bytes. Transmission rate = 2 Mb/s, vehicle speed = 5 mph, delay spread = 50 ns. Note that the lower figure is an expanded view of the upper figure.

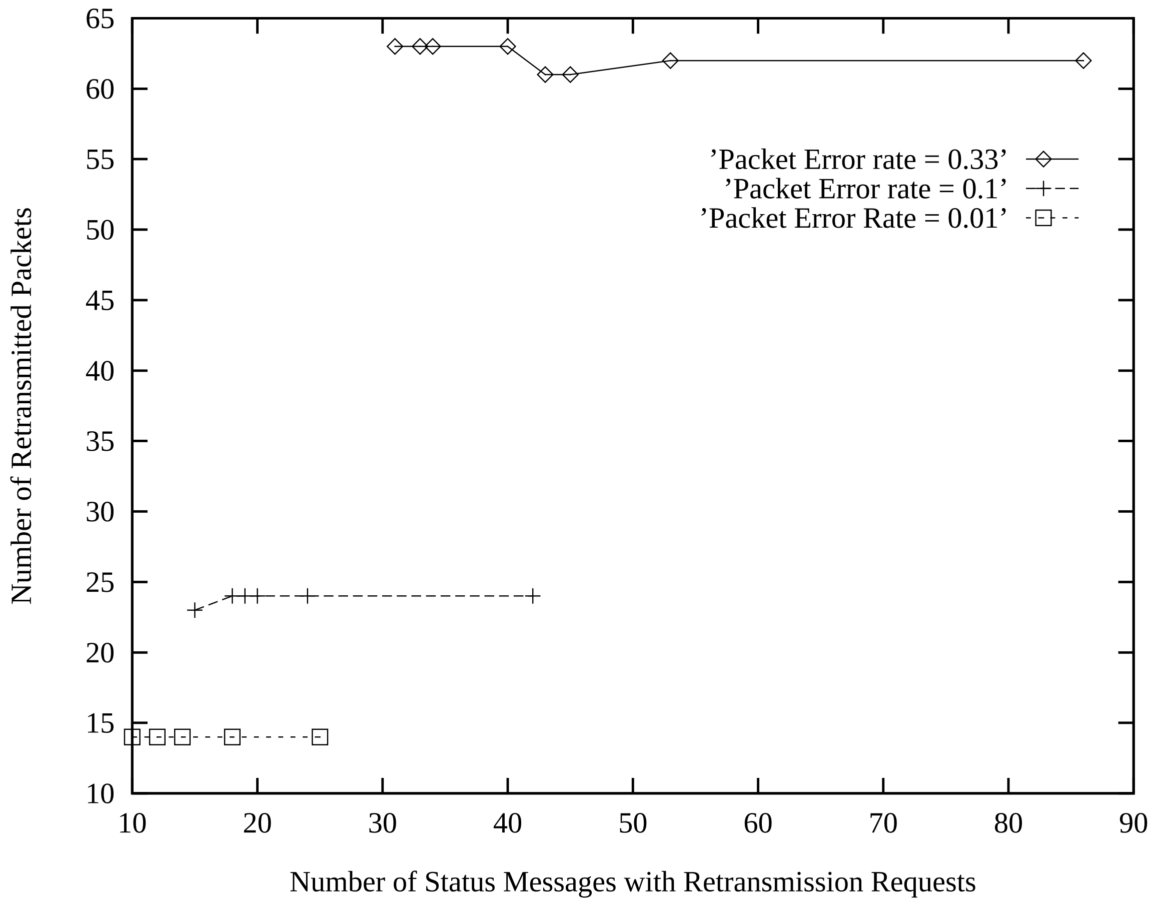


Figure 8: No redundant retransmission.

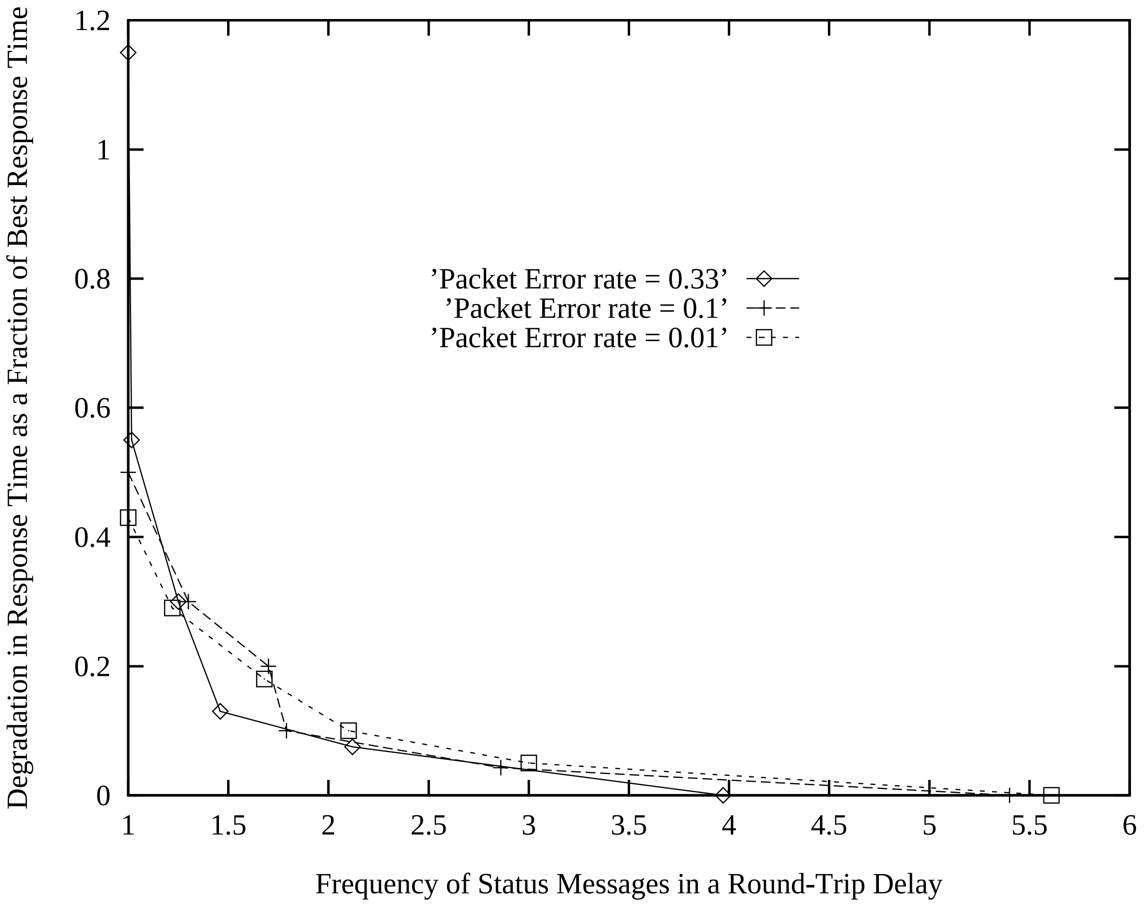
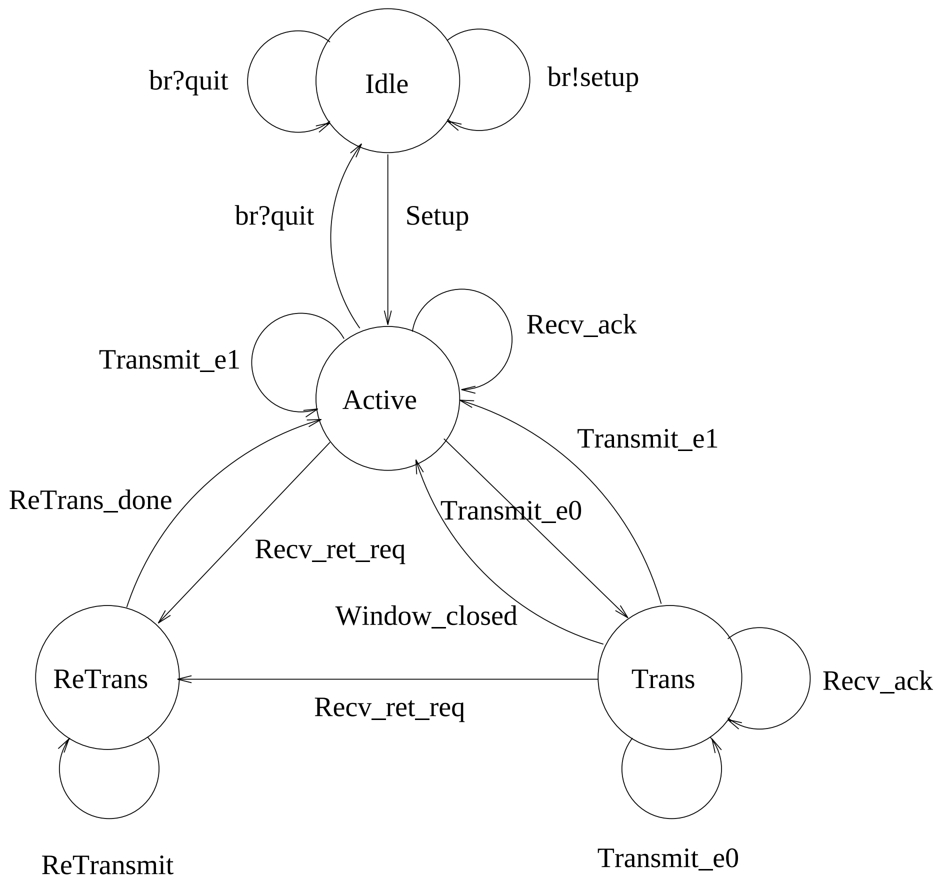
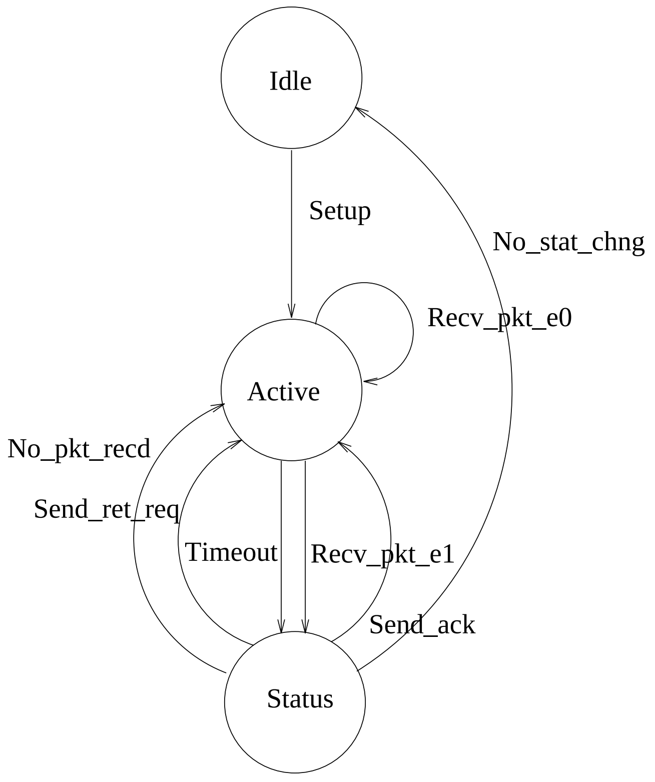


Figure 9: Trade-off between bandwidth and response time.

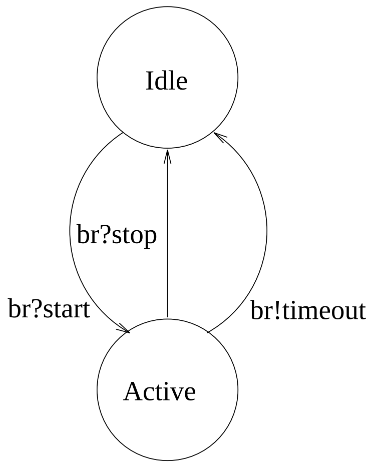


Notation	**Explanation**
Setup	br?setup_ack * u!enable [buf_ful_flag=0]
Transmit_e0	{nop < MaxBuf} u?pkt(e=0) [++Ht;++nop;update T_array] br!pkt(Ht,e=0)
Transmit_e1	u?pkt(e=1) [++Ht;++nop;update T_array] br!pkt(Ht,e=1)
Window_closed	{nop==MaxBuf} u!disable [buf_ful_flag=1]
Recv_ack	br?stat_ack(Lr) [Lt=Lr;nop=Ht-Lt;update T_array; if (buf_ful_flag==1) (buf_ful_flag=0;u!enable)]
Recv_ret_req	br?stat_ret(Lr,w,Bitmap[]) [Update T_array,Lt,nop; i=0]
ReTransmit	{((local_clock - T_array[Lr+i].clock) > rtd) AND (Bitmap[i]==0) AND (i < w)} br!pkt(Lr+i,T_array[Lr+i].ebit) [Update T_array;++i]
ReTrans_done	{(i==w) OR (T_array[Lr+i].ebit==1)}

Figure 10: Specification of Mobile Transmitter.



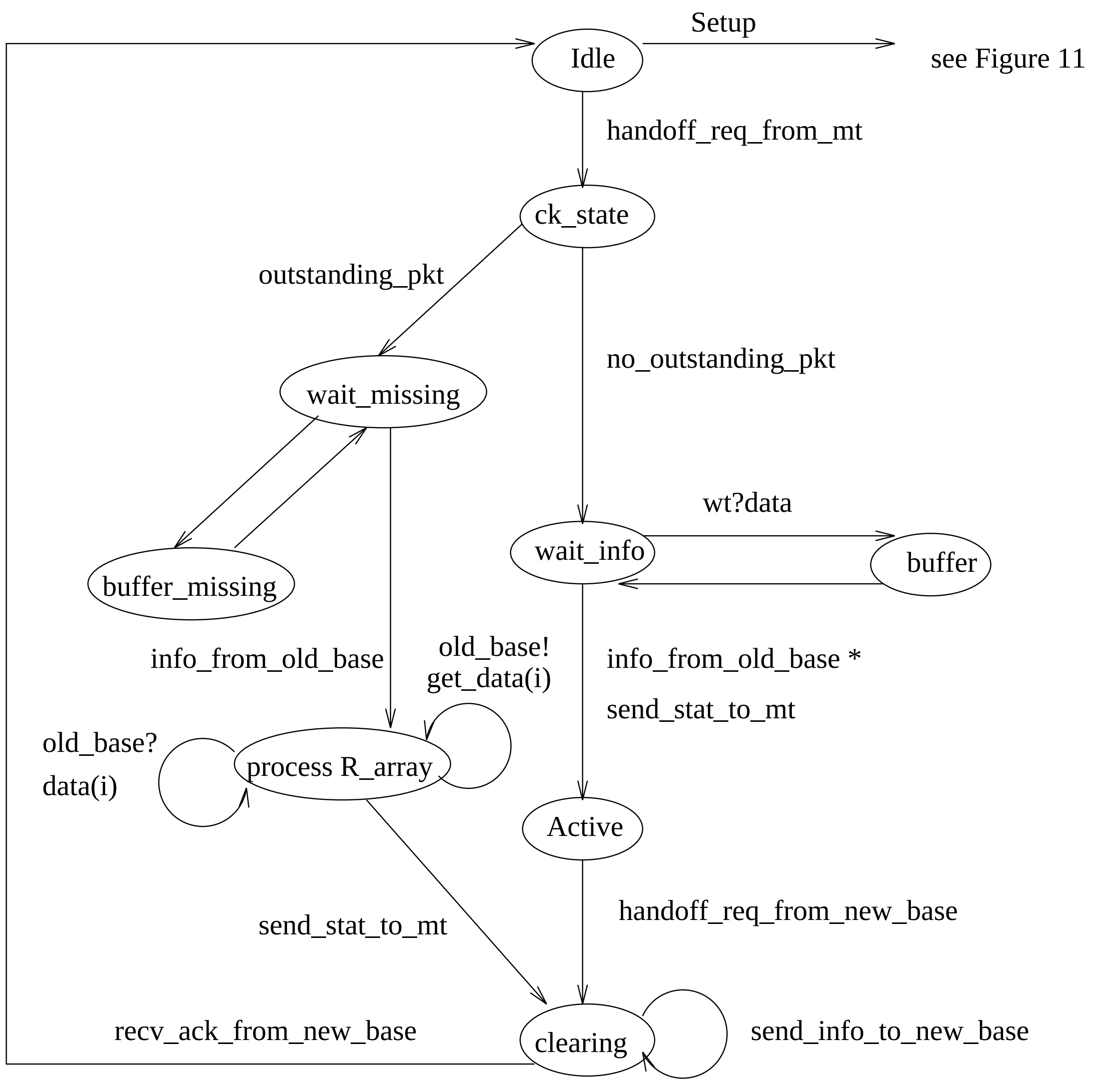
Base Receiver



Status Timer

Notation	Explanation
Setup	mt?setup * mt!setup_ack * st!start
Recv_pkt_e0	mt?pkt(pno,e=0) [if (new_info) then update R_array,Lr,Hr;nochng=0]
Recv_pkt_e1	mt?pkt(pno,e=1) * st!stop [if (new_info) then update R_array,Lr,Hr;nochng=0]
Timeout	st?timeout [++nochng]
Send_ack	{Lr==Hr+1} mt!stat_ack(Lr) * st!start
Send_ret_req	{Lr < Hr+1} mt!stat_ret(Lr,w,Bitmap[])*st!start
No_pkt_recd	{No packet received} mt!setup_ack*st!start
No_stat_chng	{nochng==MaxNochng} mt!quit

Figure 11: Specification of Base Receiver.



Notation	Explanation
handoff_req_from_mt	mt?ho_rqst(old_base,L_t,H_t) * old_base!ho_ind
outstanding_pkt	$\{L_t < H_t\}$
no_outstanding_pkt	$\{L_t = H_t\}$ [$H_r = H_t$, $L_r = L_t - 1$]
info_from_old_base	old_base?info(L_r, H_r, R_array, nochng, Maxnochng) [set variables]
send_stat_to_mt	wt!status * old_base!ack
handoff_req_from_new_base	new_base?ho_ind * new_base ! send_info(L_r, H_r,nochng,Maxnochng,R_array)
send_info_to_new_base	new_base?get_data(i) * new_base!send_data(i)
recv_ack_from_new_base	new_base?ack

Figure 12: Specification of Base Receiver during Handoff.