Anticipatory Wireless Bitrate Control for Blocks

Xiaozheng Tie, Anand Seetharam, Arun Venkataramani, Deepak Ganesan, Dennis L. Goeckel
Computer Science Department, Electrical & Computer Engineering Department
University of Massachusetts Amherst
{xztie, anand, arun, dganesan}@cs.umass.edu, goeckel@ecs.umass.edu

ABSTRACT

We present BlockRate, a wireless bitrate adaptation algorithm designed for *blocks*, or large contiguous units of transmitted data, as opposed to small packets. Our work is motivated by the observation that recent research results suggest significant overhead amortization benefits of blocks. Yet state-of-the-art bitrate algorithms are optimized for adaptation on a per-packet basis, so they can either have the amortization benefits of blocks or high responsiveness to underlying channel conditions of packets, but not both.

To bridge this disparity, BlockRate employs multiple bitrates within a block that are predictive of future channel conditions. In each feedback round, BlockRate uses a history-based scheme to predict the SNR for packets within the next block. In slow-changing scenarios as under pedestrian mobility, BlockRate uses a simple linear regression model to predict the SNR trend over the next block. In fast-changing scenarios as under vehicular mobility, BlockRate uses a path loss model to capture more significant SNR variations within a block. We have implemented a prototype of BlockRate in a commodity 802.11 driver and evaluated it via deployment on an indoor mesh testbed as well as an outdoor vehicular testbed. Our evaluation shows that BlockRate achieves up to 1.4× and 2.8× improvement in goodput under indoor and outdoor mobility respectively.

1. INTRODUCTION

Bitrate control is a critical component of a wireless protocol stack. Wireless bitrate control seeks to control the effective rate of transmission by adapting the amount of redundancy in transmitted data to the underlying channel quality so as to optimize the received goodput. Although model-based bit rate adaptation in response to channel measurements has been widely considered by the PHY community [8, 4], in recent times a variety of bitrate control schemes [13, 11, 20, 19, 21, 14] that employ measurements of the current protocol operating performance have been introduced and experimentally-verified. They are based on metrics of channel quality ranging from packet loss rate [21, 14], packet transmission time [13], signal-to-noise ratio (SNR) [11, 15], symbol-level dispersion [19], bit error rate and PHY-layer hints [20], etc.

Our work is motivated by the growing disparity between state-of-the-art bitrate control algorithms that are optimized to react on a per-packet basis and technology trends that suggest significant performance benefits to amortizing overhead across blocks, or large chunks of contiguous data transmitted as a single unit, consisting of many packets. For example, Li et al [12] demonstrate significant gains in reliable goodput using blocks by reducing the overhead of acknowledgments, timeouts, and backoffs at the link layer and redundant acknowledgments at the transport layer compared to per-packet TCP. Widely deployed commodity 802.11n cards already enable large opportunities of uninterrupted transmission (of up to 64 KB [5]) consisting of many packets. However, state-of-the-art bitrate control algorithms in research as well as in practice continue to be designed with per-packet adaptation in mind. If used as-is with blocks, these algorithms are prone to be unresponsive to changes in underlying channel quality as large blocks imply a commensurately large delay in obtaining feedback about channel quality.

This disparity raises two natural research questions that form the focus of this paper. First, do the performance benefits of large blocks outweigh the performance loss due to the reduced responsiveness of bitrate control to changes in the underlying channel quality? Second, and more importantly, is it possible to have the performance benefits of blocks without compromising on the responsiveness of bitrate control?

Our measurement-based experiments with static as well as mobile scenarios answer the first question in the affirmative. Our results show that traditional bitrate control algorithms designed to operate on a per-packet basis achieve moderately higher goodput when used as-is with blocks. This net performance benefit shows that there is greater value in amortizing overhead than reacting quickly to channel conditions even in dynamic settings. Furthermore, our results also show that there is room for improvement, i.e., an ideal block-based bitrate control scheme with future knowledge significantly outperforms packet-based bitrate control schemes used as-is with blocks, thereby setting up the stage for the second question.

Our main contribution, the design and implementation of BlockRate, a block-based bitrate control algorithm, affirms the second question as well. The key insight in BlockRate is to use multiple bitrates across packets within a block that are predictive of future chan-

ACM CoNEXT 2011, December 6–9 2011, Tokyo, Japan.

nel conditions. While this may sound impossible at first glance, our measurement experiments across several typically encountered pedestrian and vehicular mobility scenarios in indoor as well as outdoor settings show both that it is possible to predict the future SNR and that leveraging this predicted information to select the bitrate translates to significant gains in goodput. BlockRate maintains a receiver-assisted technique to maintain a mapping between the SNR and the best bitrate corresponding to that SNR. BlockRate uses this history-trained SNR-to-bitrate mapping in conjunction with its model for predicting the future SNR to pick (possibly different) bitrates for packets in the next block.

BlockRate uses two simple models to predict the SNR experienced by packets in the near future. The first model is applicable to slow-changing scenarios such as static or pedestrian mobility scenarios. In such slow-changing scenarios, BlockRate employs a *linear regression model* based on a historic time series of SNR values to predict the future SNR. The second model is applicable to fast-changing scenarios as is typical under vehicular mobility. In such fast-changing scenarios, BlockRate employs a *path loss model* in conjunction with the historic time series of SNR values and its knowledge of the distance to the receiver to predict the future SNR.

We implemented a prototype of BlockRate in the MadWiFi driver [2] and deployed it on an indoor mesh testbed and an outdoor vehicular testbed consisting of 35 vehicles moving in a 150 sq. mi area in and around Amherst, MA. We also conducted trace-driven evaluations using the ns-3 simulator[3] to compare BlockRate with state-of-the-art algorithms relying on PHY or sensor hints. Our evaluation shows that, compared to existing packet-level schemes, BlockRate achieves up to 1.4× and 2.8× improvement in goodput in pedestrian and vehicular mobility scenarios respectively.

2. A CASE FOR BLOCK BASED BITRATE CONTROL

In this section, we experimentally motivate the need for block-based bitrate control. Our measurement-based experiments reveal two findings. First, the performance benefits of amortizing overhead with blocks outweigh the performance loss due to less responsive bitrate control resulting in a net gain. Second, an ideal block-based protocol with future knowledge can significantly improve upon this gain compared to using existing packet-based bitrate control schemes as-is with blocks. We begin with a brief background on the benefit of blocks.

2.1 Background: Blocks versus packets

Although one would expect MAC and transport protocols to be engineered so as to ensure low overhead, that is not the case in practice today. To appreciate this, consider Figure 1 that shows a comparison of the transmission overhead associated with sending a packet and a block in 802.11a/b/g networks. Acquiring a transmission opportunity entails a carrier sensing interval (DIFS) and a possible backoff. A successful receipt incurs a brief holding period (SIFS) and an acknowledgment transmission time. Under good channel conditions, this combined overhead is acceptable. However, poor channel conditions may latch on several rounds of

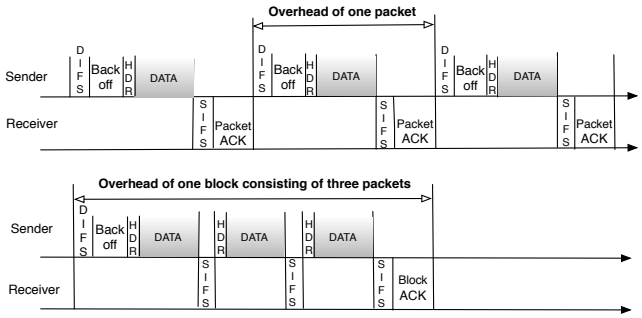


Figure 1: Overhead for packets and blocks. A block exceeding the MAC transmission opportunity limit may be interspersed by multiple DIFS rounds (not shown).

sender timeouts and backoffs of increasing length until a successful transmission. Li et al. [12] show that for reliable transport, cross-layer effects in the form of redundant TCP acknowledgments and negative interactions between TCP rate control and fluctuating link- and network-layer delays further exacerbate the net impact of using small packets as a unit of transmission.

Thus, it is important to amortize overhead by sending as much as possible in each transmission opportunity. Simply sending massive packets is impractical as the likelihood of corruption exponentially increases with the packet size and network-layer MTUs limit the size of packets. Instead, *blocks* consisting of multiple packets have come to be recognized as a better alternative. For example, as illustrated in Figure 1, widely used 802.11n implementations allow for blocks of up to 64 KB [5] in each transmission opportunity. Li et al. [12] advocate blocks of up to 1MB with multiple carrier sensing rounds within each block so as to keep delays for competing interactive connections small.

To illustrate the overhead benefits of using 802.11n-style blocks, consider a back of the envelope calculation based on the values measured by the MadWiFi driver [2]. Assume packets of size 1.5KB and a block comprising of just *three* such packets as shown in Figure 1. The DIFS, SIFS and packet header have transmission times of 28μs, 9μs and 20μs respectively. For packet transmissions, the acknowledgement takes 200μs while the average backoff duration is 436μs; the same values for transmitting a block are 220μs and 528μs[1]. The overhead, i.e., the sum of DIFS, SIFS, header, backoff and acknowledgement, for a bitrate of 12Mbps of transmitting a block is a factor 2.3× lower compared to packets. This reduced overhead in turn translates to a 1.3× improvement in goodput.

2.2 Bitrate control in blocks

Although large blocks improve performance by amortizing overhead, there is an obvious downside, namely that it severely reduces the responsiveness of bitrate control as feedback about channel quality is delayed by a factor proportional to the size of a block. Commodity 802.11n cards using bitrate control schemes primarily designed for per-packet adaptation attempt to balance this trade-off by simply limiting the block size. The

[1]These values were obtained based on measurements on a local mesh testbed as detailed in §4

block-based transport protocol, Hop [12], does not address the problem of rate control, experimenting primarily with a fixed hand-picked bitrate. More recent bitrate control schemes optimized for 802.11n [14] leverage the MIMO nature of its PHY layer, but continue to implicitly assume per-packet rate adaptation.

This state-of-the-art leaves open the question of whether existing bitrate control schemes designed with per-packet adaptation in mind are adequate if used as-is with blocks and if not, how much room is there for improvement? We study this question in an indoor static setting and an outdoor vehicular setting in turn next.

2.3 Blocks vs. packets: Static indoor

We conduct a simple experiment on an indoor 802.11g mesh testbed of 16 nodes (Figure 7) to compare the effectiveness of bitrate control using blocks and packets. We randomly pick 30 links and continuously send UDP packets, selecting one link at a time. The packet size is 1.5KB and each block contains 200 packets. A more detailed description of the testbed is deferred to §4.

We compare the performance of two block-based bitrate control schemes, Oracle+Block and Charm+Block, against two packet-based ones, SampleRate [13] and Charm [11], across different links in the testbed. Oracle+Block picks the bitrate that is known through recent measurements to be the best bitrate for that link. Oracle+Block is intended to serve as a lower bound on the performance achieved by an optimal bitrate control scheme. Charm+Block uses the Charm bitrate control algorithm [11] as-is in conjunction with blocks. Charm and SampleRate are both well-known packet-level bitrate control schemes that have been shown to work well in 802.11a/b/g networks; the former adapts the bitrate based on the SNR while the latter adapts the bitrate based on the time to transmit a packet successfully.

Figure 2 shows the results for the compared bitrate control algorithms, where each line is the CDF across all links of the goodput achieved by that algorithm. The experiment yields two important insights. First, the block-based algorithms, Oracle+Block and Charm+Block, significantly outperform the corresponding packet-based algorithms. In particular, Oracle+Block achieves a median goodput improvement of 2.1× compared to Charm, which results from a reduction in the effective per-packet transmission time from 2.3ms for Oracle+Block to 1.1ms for Charm. Second, in static indoor settings, simply using a packet-based algorithm as-is with blocks, as indicated by Charm+Block, yields significant gains but leaves some room for improvement. This is unsurprising as the gains of amortizing overhead using blocks significantly outweigh the loss due to reduced responsiveness of bitrate control; in an indoor static setting, the channel quality does not change enough to warrant frequent step-ups or step-downs in the bitrate.

Figure 3 shows that the goodput gains result from the bitrates selected by the different schemes. While Oracle+Block selects a fixed bitrate for each link, SampleRate and Charm select bitrates for every packet dynamically depending on their assessment of the channel quality. We find that both SampleRate and Charm select a single bitrate nearly 70% of the time, as expected in a static indoor setting. We call this bitrate the *major*

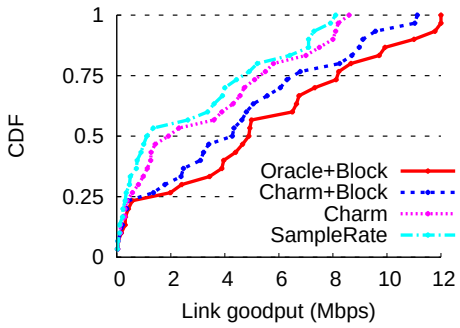


Figure 2: Static indoor: Oracle+Block achieves a 2.1× higher median goodput than Charm that monitors SNR on a per-packet basis.

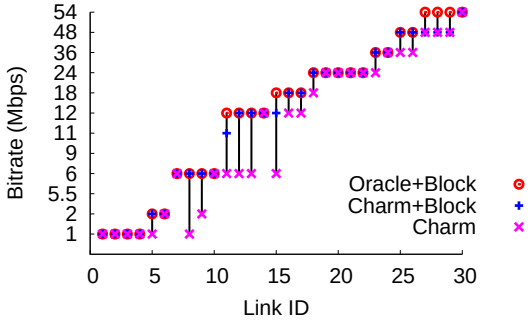


Figure 3: Static indoor: Oracle+Block selects higher bitrates than Charm on 16 links corresponding to the vertical lines. Each vertical line connects bitrates chosen by the three schemes when they are unequal.

bitrate. As SampleRate happens to select the same major bitrate as Charm for every link, we omit it in Figure 3 for clarity. The figure shows that Oracle+Block selects a higher bitrate than SampleRate and Charm for 16 out of 30 links while Charm+Block does so on 13.

Why do block-based schemes pick a higher bitrate compared to packet-based schemes? In particular, why does Charm+Block pick a higher bitrate than Charm (over packets)? To appreciate this, recall that all bitrate control algorithms fundamentally seek to maximize goodput, or equivalently $\frac{1-l(r)}{t(r)}$, where $t(r)$ is the average time (including overheads such as carrier sensing, timeouts, acknowledgments, backoffs, etc.) for each transmission of a packet and $l(r)$ is the loss rate at the bitrate r. Consider two bitrates r_1 and r_2 ($r_1 < r_2$). For packet-based schemes, suppose the corresponding loss rates are, say, $l(r_1) = 0.1$ and $l(r_2) = 0.3$, and packet transmit times (including overhead) are $t(r_1) = 10$ and $t(r_2) = 9$ respectively. Packet-based schemes will select the smaller bitrate r_1 as $\frac{1-l(r_1)}{t(r_1)} > \frac{1-l(r_2)}{t(r_2)}$.

In comparison, blocks reduce the effective overhead associated with each packet transmission. Suppose the average transmit time of each packet sent as part of a large block is $\tau(r_1) = 8$ and $\tau(r_2) = 5$ at the bitrates r_1 and r_2 respectively. Then, a block-based scheme will select the higher bitrate r_2 as $\frac{1-l(r_1)}{\tau(r_1)} < \frac{1-l(r_2)}{\tau(r_2)}$. Note that in this example, $\tau(r_2)$ decreased by a bigger factor (from 9 to 5) compared to $\tau(r_1)$ (from 10 to 8), which is consistent with the observation that blocks reduce the

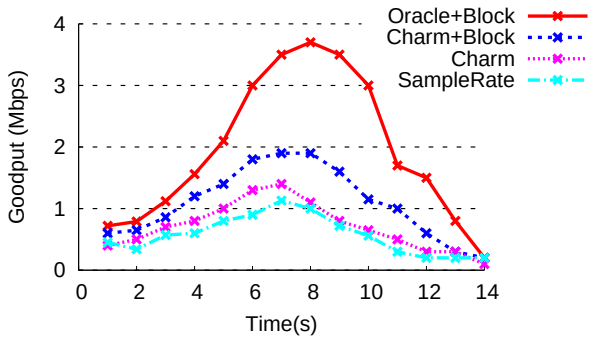


Figure 4: Vehicular mobility: Oracle+Block achieves 2× higher goodput than Charm+Block.

overhead significantly in poorer conditions, i.e., when loss rates are higher. Thus, blocks enable bitrate control to explore higher (more lossy) bitrates aggressively without fear of a steep increase in loss-induced overhead.

2.4 Blocks vs. packets: Vehicular outdoor

The results above might appear to suggest that existing bitrate control algorithms used as-is with blocks achieve much of the possible performance gains with a small room for improvement. However, the situation is very different in mobile settings with frequent and significant variations in channel quality.

To assess the impact of mobility, we conduct an experiment involving vehicular mobility in an outdoor setting. This experiment involves a laptop placed in a car driven by one of the authors sending back-to-back UDP packets to another stationary laptop acting as an access point. The packet and block sizes are the same as in the static, indoor experiment described above.

Figure 4 shows the goodput achieved by the compared bitrate control algorithms as a function of time. All schemes show a period of increasing goodput followed by decreasing goodput, which is consistent with the vehicle first moving towards the access point and then away from it. As before, the two block-based schemes, Oracle+Block and Charm+Block, significantly outperform packet-based schemes. However, unlike the static indoor setting, we observe a significant difference (of up to 2×) in the goodput achieved by Oracle+Block compared to Charm+Block, i.e., a well-designed block-based bitrate control scheme significantly outperforms a packet-based scheme used as-is with blocks.

The rest of the paper describes a practical block-based bitrate control scheme that achieves a goodput comparable to Oracle+Block in a variety of mobile scenarios. Above, Oracle+Block itself was obtained using a tedious a priori measurement process where one of the authors experimented with all possible bitrates at points spaced 5 meters apart from the access point. The measurements at each point were conducted while remaining stationary; for this reason as well as possible temporal variations in channel quality, we intend for Oracle+Block to serve as a lower bound on the ideal goodput possible using blocks.

Summary: The performance gains obtained by amortizing overhead using large blocks more than outweigh the loss due to reduced responsiveness of bitrate control to the underlying channel quality in static as well as in mobile scenarios. However, in mobile scenarios, state-of-the-art bitrate control schemes used as-is with blocks leave significant room for improvement compared to an ideal scheme with future knowledge, making the case for a bitrate control scheme optimized for blocks.

3. BlockRate DESIGN

In this section, we present BlockRate, a block-based bitrate control algorithm that achieves high goodput in a variety of static and mobile settings. The key insight in BlockRate is to use multiple bitrates across packets within a block that are selected based on the predicted SNR trend over the transmission of the block.

3.1 Overview

BlockRate uses the SNR measured at the receiver to learn for each SNR regime and bitrate the corresponding packet loss rate. The receiver learns this mapping, referred to as the *SNR-bitrate table*, by monitoring the loss rates of packets sent at various attempted bitrates by the sender in the recent past. The receiver uses this table to select the best bitrate for each packet in the next block based on the predicted SNR for the packet, and conveys this information to the sender piggybacked with a selective acknowledgment for packets within the current block.

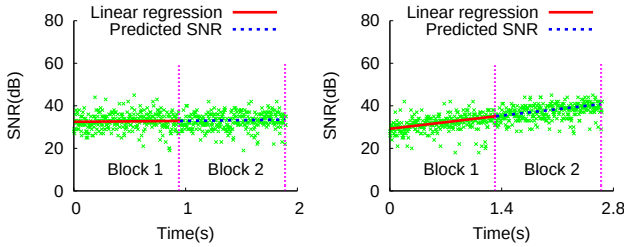
The high-level design described above is similar to existing SNR-based bitrate control schemes [11, 15], but with an important difference in the prediction step. Packet-based bitrate control schemes simply assume that channel conditions do not change significantly in the course of a packet or two, so the SNR predicted for the next packet is the same as that of the current packet. However, as shown in the previous section, this simplistic prediction performs poorly at the block granularity.

To address this problem, BlockRate uses two simple models to predict the SNR for packets within the next block. The first is a *linear regression model* invoked in slow-changing environments such as in static or pedestrian mobility scenarios. The second is a *path loss model* invoked in fast-changing scenarios such as under vehicular mobility. We explain each of these in detail next.

3.1.1 Linear regression model

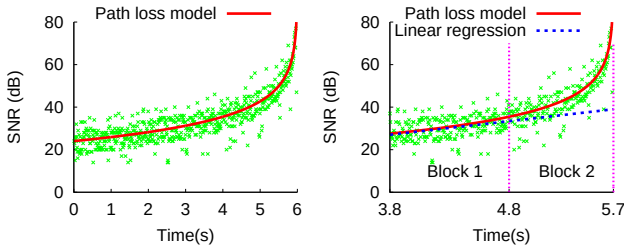
In static or pedestrian mobility scenarios, the SNR trend changes slowly. To appreciate this, consider Figures 5 showing the variation in packet SNRs from one block to the other in an indoor setting, for the static and the pedestrian mobility scenarios respectively. To measure the SNR, we use two laptops configured in a sender-receiver mode. In the first experiment (Figure 5(a)), both laptops are kept static (with no line of sight), while in the second one (Figure 5(b)) of them is kept stationary with the other being moved towards it at walking speeds.

Figure 5(a) shows that in the static scenario, the average SNR remains constant at short time scales with most fluctuations confined to a range of 10 dB. Figure 5(b) shows that in the walking case, the fluctuations persist and are accompanied by a weak upward trend as the receiver moves towards the sender. Thus, if the pedestrian user's speed does not change significantly over the course of a couple block transmissions,



(a) Static (b) Pedestrian

Figure 5: SNR variation: static and pedestrian mobility.



(a) Vehicular (b) Vehicular (zoomed)

Figure 6: SNR variation: vehicular mobility.

the trend inferred from recent blocks can be used to predict the SNR for the next block. For a maximum block size of 300KB (the size used throughout this paper) we find that block transmission times are typically between 0.2 to 1.5 seconds. As a result, we find that the predictable SNR trend at time scales of block transmission times, as shown in the example in Figure 5(b), is characteristic of our entire set of pedestrian mobility traces (not shown for brevity).

In such slow-changing scenarios, a simple linear regression model [1] can effectively capture SNR trends across consecutive blocks. The model assumes that the received SNR has a simple linear relationship with time, and fits a straight line through the set of SNR samples using the least squares estimate. BlockRate extends this regression line to predict the SNR variation over the course of the next block.

3.1.2 *Path loss model*

In outdoor vehicular mobility scenarios, SNR variations can be rather steep within the course of a block, as the vehicle's distance from the access point can change significantly during that time. For example, Figure 6(a) shows the time series of packet SNRs observed by a receiver kept in a car approaching a sender (AP) at 30 mph. We conducted this experiment in a downtown area using two laptops with the sender laptop placed at a fixed location by the road. The figure shows that the SNR changes rapidly within several seconds and, more importantly, does not show a simple linear relationship with time (especially in the 3–6 second regime).

Figure 6(b) further zooms into the SNR of packets within two consecutive blocks in this regime. The figure shows that the linear regression line deviates considerably from the actual SNR, so any estimation based on linear extrapolation will likely result in suboptimal performance. More importantly, the SNR shows this sharp, non-linear trend when the vehicle is closest to the AP and has a good channel (SNR $>$ 50 dB). We show in §4.2 that the highest bitrates and goodput are achieved during this time period, so it is critical to capture these sudden changes in SNR. One approach to alleviate this problem might be to reduce the block size so that the linear regression method can continue to be reasonably accurate. However, we find that this significantly impairs the overhead amortization benefits of blocks, thereby hurting goodput.

To address this problem, BlockRate uses the wireless channel path loss model [16] to model the non-linear variation of SNR over time. The model postulates a logarithmic relationship between path loss and distance:

$$PL(d) = PL(d_0) + 10\alpha \log_{10}(d/d_0) \quad (1)$$

where PL is the path loss in dB, i.e., PL = transmit power − receive power, d_0 is the reference distance, and α is the path loss exponent. Since on a dB scale, SNR = transmit power − PL − noise, assuming that the transmit power at the sender and the noise at the receiver are fixed, we can rewrite Eq. 1 as follows:

$$\text{SNR}(d) = \text{SNR}(d_0) - 10\alpha \log_{10}(d/d_0) \quad (2)$$

i.e., SNR increases logarithmically with decreasing distance. Figure 6 shows that the path loss model indeed fits the SNR variations under vehicular mobility well.

To use the path loss model for SNR prediction, BlockRate needs to know the distance d between the sender and the receiver. To this end, we make two assumptions. First, we assume that GPS devices are readily available in vehicles, and both the sender and the receiver periodically broadcast beacons containing location information. Second, we assume that vehicles move with constant speed, i.e., the distance between the sender and receiver changes at a uniform rate, over the course of successive block transmissions.

With the assumptions above, BlockRate uses the path loss model to predict the future SNR as follows. The reference distance d_0 is chosen to be 50m and beacon messages are used to estimate d periodically. Consider two successive block transmissions and let d_1 and d_2 be the distances (calculated using the GPS beacons) at the beginning of the first and second blocks respectively. At first, the receiver uses these distances and the SNR of received packets to estimate the path loss exponent. Assuming uniform motion, the distance between the sender and receiver at the end of the second block is predicted to be $|2d_2 - d_1|$. Knowing this distance variation and the estimated path loss exponent, BlockRate can estimate the SNR variation for the next block using Eq 2. A moving average of the path loss exponent is updated with each received GPS beacon.

The path-loss model's dependence on location information currently limits its applicability. Although mobile devices are increasingly equipped with an inexpensive GPS, their location estimates can be error-prone, causing BlockRate to incorrectly estimate the instantaneous path loss exponent. Nevertheless, we find that a moving average of the path loss exponent is sufficiently robust to small errors. The estimated SNR in Figure 6 (as well as in all experiments involving BlockRate under vehicular mobility in this paper) incorporates the effect of GPS errors observed in a variety of outdoor settings.

Bitrate (Mbps)	**1**	$\cdots$	**12**	**18**	**24**	$\cdots$	**54**
`rcvd`	0	$\cdots$	72	70	54	$\cdots$	0
`lost`	0	$\cdots$	11	19	42	$\cdots$	0

Table 1: SNR-Bitrate table (SNR=45dB).

Bitrate (Mbps)	**1**	$\cdots$	**12**	**18**	**24**	$\cdots$	**54**
Rcvd	0	$\cdots$	72	70	54	$\cdots$	0
Lost	0	$\cdots$	11	19	42	$\cdots$	0

Table 2: SNR-Bitrate table (SNR=45dB).

A portion of our vehicular mobility experiments also involve a moderate amount of acceleration (as is naturally observed in the motion of public transit vehicles), suggesting that BlockRate's simplistic assumption of uniform motion for the length of a block transmission time works well in practice.

Looking further out, we think that it may be possible to eliminate the need for a positioning system in BlockRate. To this end, more sophisticated algorithms to predict channel conditions under various assumptions [7] could be used. It may also be feasible to jointly estimate both the distance as well as the path loss exponent using more sophisticated multivariate regression techniques. Our primary goal in this paper is to establish the feasibility of anticipatory bitrate control using readily available information on mobile devices today (similar in spirit to [17]), so further refinement of BlockRate's prediction technique or reducing its reliance on GPS is deferred to future work.

3.2 Building the SNR-Bitrate mapping table

An SNR-bitrate table is used to select the best bitrate at each SNR. It records the number of received and lost packets at all available bitrates in each SNR regime. Table 2 shows an example of the mapping information at an SNR of 45 dB.

The receiver updates the SNR-bitrate table for each received block. For a correctly received packet in the block, the receiver increments the `rcvd` field in the table based on the packet's SNR and bitrate. For a lost packet, the receiver estimates its SNR and bitrate as the SNR and bitrate of the most recent correctly received packet and increments the `lost` field using these values. The table keeps data obtained in a recent window of five seconds as older values yield little additional benefit

Our estimation of the SNR for the lost packet is based on the assumption that the channel SNR remains unchanged over the course of a small number of packet transmissions. This helps us avoid short-sighted reaction to interference or unpredictable short-term changes in channel quality. Similar assumptions have been made by other SNR-based bitrate control schemes [11].

3.3 Selecting the bitrate

Given the SNR-bitrate table, BlockRate selects the bitrate that maximizes the goodput at each SNR. The goodput is computed as $b*(1-l)$, where b is the bitrate and l is the lossrate. In the previous example in Table 2, the bitrates of 12Mbps, 18Mbps and 24Mbps have loss rates of 13%, 21%, 44% respectively, so the corresponding goodputs are 10.4Mbps, 14.2Mbps, and 13.4Mbps respectively. Thus, 18Mbps is the best bitrate for an SNR value of 45 dB.

We observed from our experiments that a continuous range of SNR values tend to be mapped to the same best bitrate in a given environment. Hence, as the packets within a block have gradually changing SNR, the best bitrate for the packets should change following a step pattern, i.e., the best bitrate remains the same for a certain number of packets, jumps to the next higher or lower one, and again persists for some packets.

BlockRate uses the above observation to optimize transmission overhead. Once the receiver selects the best bitrates for packets in the next block based on their predicted SNR, it compacts this information using a runlength-encoded map. For example, assume that a block has 200 packets and packets numbered between 1-115 use 12Mbps while those between 116 -200 use 18Mbps. The receiver then constructs a table specifying just the range of packets and the bitrates to be used, in this case [115, 12; 200,18]. This compact table is then inserted into the block acknowledgement and sent back to the sender.

Suppose at this point that the selected set of best bitrates for packets in the next block are {18Mbps, 24Mbps}. In addition to sending packets at the receiver-recommended bitrates, BlockRate also probes bitrates one level lower and higher in order to detect better transmission opportunities. BlockRate sends 10% of the total packets within a block at each of the two probe bitrates. In the above mentioned example, the sender then randomly picks 10% packets within the block to send at 12Mbps, and another 10% to send at 36Mbps. We observe that a typical block transmission contains 3 to 4 different bitrates including the probed bitrates in an indoor environment, and 4 to 5 in an outdoor setting. This simple probing policy captures sufficient bitrate diversity so as to enable the quick discovery of better transmission opportunities.

3.4 Implementation

We implemented BlockRate primarily in the MadWiFi driver [2] as follows. BlockRate's block transfer protocol is a simple, unreliable single-round protocol (unlike the reliable multi-round protocol in Hop [12]). The sender transmits all packets in a block with link-layer acknowledgments disabled. To further reduce overhead, we set the transmission opportunity (txop) to its maximum value. A txop is a contention-free burst of link-layer packets separated by SIFS and its maximum value allowed in 802.11e is 8192 microseconds (or about 35 1.5KB packets at a bitrate of 54 Mbps). A transmitter has to perform carrier sense after each txop, so it does not hog the channel for the entire duration of a block transmission. The block sized used throughout this paper is 300KB, i.e., 200 packets of size 1.5KB each.

The receiver sends a block acknowledgment back to the sender at the end of a block transmission, as identified by a small "reserved" packet that is always transmitted at the lowest bitrate to maximize the likelihood of its delivery. The receiver gathers the received signal strength indicator (RSSI) values for each received packet as reported by the driver and conveys them to the sender via the block acknowledgment. The sender considers these RSSI values as equivalent to the received SNR, in accordance with the MadWiFi documentation

[2]. As we were unable to implement the block acknowledgement in the kernel space of the MadWiFi driver, we designed a user-space module for sending the block acknowledgement. When the acknowledgement is received at the sender, it uses *sysctl* to pass all the required information to the driver so that it can process the next block. Except for this block acknowledgment, all other components of BlockRate including the rate selection algorithms above are implemented in the driver.

In our experiments, BlockRate obtains location information using USB GPS devices installed on computers in our vehicular testbeds. The devices broadcast GPS information once every second. Currently, we manually configure BlockRate to use one of the two predictive models (i.e., linear regression or path loss) depending on whether the node is engaged in static/pedestrian or vehicular mobility in the experiment. Although we have not implemented automatic detection of a user's mode of mobility, it is straightforward to classify mobility as vehicular using a reasonable speed threshold (e.g., > 4 meters/sec). As speed is the primary factor that determines which model is appropriate, we expect that such a simple policy would suffice in practice.

4. EVALUATION

In this section, we compare BlockRate to existing packet-based bitrate adaptation schemes as well as adaptations of these schemes for blocks. A summary of our results is as follows.

- BlockRate improves goodput over a packet-level SNR-based scheme by up to 2.8× under vehicular mobility (§4.2) and by up to 1.4× under pedestrian mobility (§4.3 and §4.6).
- BlockRate's anticipatory design helps improve goodput by up to 1.6× under vehicular mobility over an SNR-based scheme used as-is with blocks (§4.2).
- BlockRate retains significant goodput improvements under interference (§4.4); in conjunction with a reliable transport protocol (§4.5); and in comparison to packet-level bitrate control schemes relying on PHY or sensor hints (§4.6).

4.1 Experimental setup

Deployment.

We evaluate BlockRate via deployment on an indoor mesh testbed and two outdoor vehicular testbeds, referred to as MESH, VEHICULAR1, and VEHICULAR2 respectively. MESH is a wireless mesh testbed consisting of 16 nodes located in one floor of our computer science building (Figure 7). Each node is a Mac Mini computer running Linux 2.6 with a 802.11a/b/g Atheros/MadWiFi card. VEHICULAR1 is an outdoor vehicular testbed comprising of 35 public transport vehicles operating in an area of 150 square miles. Each vehicle is equipped with a Hacom OpenBrick 1GHz Intel Celeron M system running Linux 2.6 and Atheros/MadWiFi cards. The mobility of the public transport vehicles is not under our control, so in order to experiment with varying levels of mobility, we also deployed BlockRate on a private vehicular testbed, VEHICULAR2, consisting of two cars each of which is equipped with a MacBook computer and the same wireless card as MESHÀll cards are configured in ad hoc mode with RTS/CTS turned off.

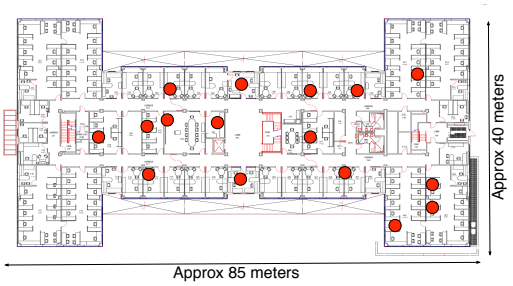


Figure 7: The MESH testbed showing node locations.

Trace-driven evaluation.

We also conducted trace-driven experiments in the ns3-simulator[3] to compare BlockRate with SoftRate[20] and RapidSample[17]. These recently proposed bitrate control algorithms rely on PHY or external movement hints and require different hardware platforms than ours. So, we use the traces from the SoftRate[20] paper which list for each 5 ms time slot the SNR, the transmission fate of packets at every bitrate, and SoftPHY hints for each packet, retaining their assumption that the channel remains invariant within a 5ms interval.

Compared bitrate control schemes.

We compare BlockRate against the following schemes.

SampleRate [13], a packet-based scheme that maximizes goodput by selecting the bitrate with the lowest time to (successfully) transmit a packet.

Charm [11], a packet-based scheme that leverages channel reciprocity to estimate SNR and uses SNR thresholds to select the best bitrate.

SoftRate [20], a packet-based scheme that uses SoftPHY hints from a modified physical layer to estimate bit error rate (BER) and select the bitrate.

RapidSample [17], a packet-based scheme designed for mobile scenarios that aggressively reduces bitrates upon losses that it estimates to be induced by mobility.

Charm+Block, a straightforward adaptation of Charm to blocks. It preserves Charm's design of using channel reciprocity to predict SNR and using SNR thresholds for selecting bitrates. However it assumes that all packets in a block have the same predicted SNR, so all packets in a block are transmitted at the same bitrate.

Oracle+Block, as in Section 2.3 and 2.4, attempts to estimate the ideal bitrate by trying each bitrate in succession at each location. Because of the effort involved in this process, we were able to measure Oracle+Block's performance only in a small number of experiments.

It is unclear how to make SampleRate work on blocks without drastically modifying its key design. SampleRate selects the bitrate based on per-packet transmission time including the backoffs and retransmissions. However, BlockRate transmits packets in a block back-to-back without retransmissions and the backoff times between txop's in a block are primarily determined by contention, not channel quality, so it is unclear how to compute an effective block transmission time that is meaningful for bitrate control.

Table 3 summarizes the experimental setup. We use

Section	Evaluation method	Environment	Setup	Testbed
§4.2	Deployed prototype	Outdoor, fast-changing	Vehicle-to-vehicle Vehicle-to-AP	VEHICULAR1 VEHICULAR2
§4.3	Deployed prototype	Indoor, slow-changing	Pedestrian Static	VEHICULAR2 MESH
§4.4	Deployed prototype	Indoor, with interference	Static	MESH
§4.5	Deployed prototype	Indoor, reliable transfer	Static	MESH
§4.6	Trace-driven	Indoor, slow-changing	Pedestrian	ns3-simulator

Table 3: A summary of the experimental setup.

	Number of contacts (one day)	Average contact duration (second)
SampleRate	1719	10.2
Charm	1524	11.8
Charm+Block	1822	8.9
BlockRate	1745	9.2

Table 4: Number of contacts and contact duration for the four algorithms in vehicle-to-vehicle experiment.

blocks of size 200 packets for all the experiments. The packet size is 1.5KB for the deployment-based experiments and 1KB for the trace-driven evaluation so that the experimental setup is consistent with SoftRate and RapidSample. All experiments measure UDP goodput except for the reliable goodput experiments in §4.5.

4.2 Outdoor vehicular mobility

We first evaluate BlockRate in fast-changing outdoor settings where communicating nodes move at vehicular speeds. We consider two typical scenarios: (1) vehicle-to-vehicle, where two vehicles communicate when they pass each other; (2) vehicle-to-AP, where a moving vehicle communicates with a static access point.

4.2.1 *Vehicle-to-vehicle* [VEHICULAR1]

We use the VEHICULAR1 testbed to compare the performance of different algorithms under vehicular mobility. Two vehicles communicate with each other and exchange data when they cross one another. As it is not possible to run all algorithms simultaneously with a single NIC, we execute each algorithm on all vehicles for one weekday and analyze the aggregate data. Although the vehicular mobility patterns vary from one day to the other, we observe from Table 4 that the number of contacts and average contact duration for all the algorithms are more or less comparable. For example, the number of contacts for all algorithms is between 1500 and 1800 and the average contact duration is between 9 and 12 seconds. Furthermore, as we are measuring the average goodput, not the volume of data transferred, the small differences in the number or average duration of contacts only introduce a weak sampling bias.

Figure 8 shows the CDF of the goodput across all vehicle-to-vehicle contacts for each of the four algorithms. We observe that BlockRate achieves a median goodput of 1.14Mbps, a 1.3× improvement over Charm+Block and at least 1.6× over SampleRate and Charm.

The improvement over Charm+Block attests to BlockRate's anticipatory design, i.e., its ability to predict the future SNR trend and select multiple bitrates within a block. Charm+Block performs worse because it uses the same bitrate for all packets in a block and is unable to adapt to intra-block variations in channel quality. We note that Charm+Block does outperform SampleRate and Charm, showing the overhead amortization benefits of blocks over packets. Taken together, these results show that the gains of amortizing overhead using blocks help all schemes (despite the reduced responsiveness), however an anticipatory block-based bitrate control scheme significantly outperforms packet-based schemes used as-is with blocks.

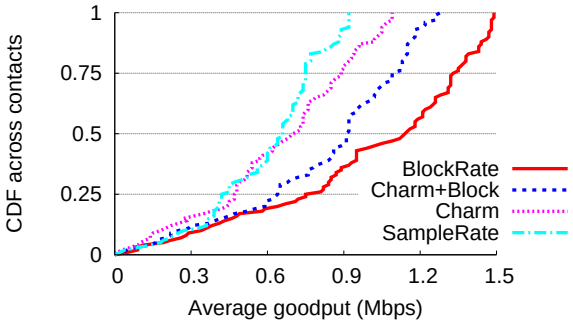


Figure 8: CDF of goodput in vehicle to vehicle experiment. BlockRate improves median goodput by 1.3× over Charm+Block and 1.6× over SampleRate and Charm.

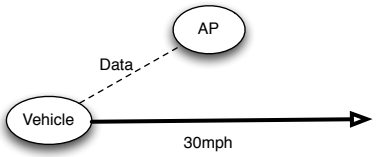
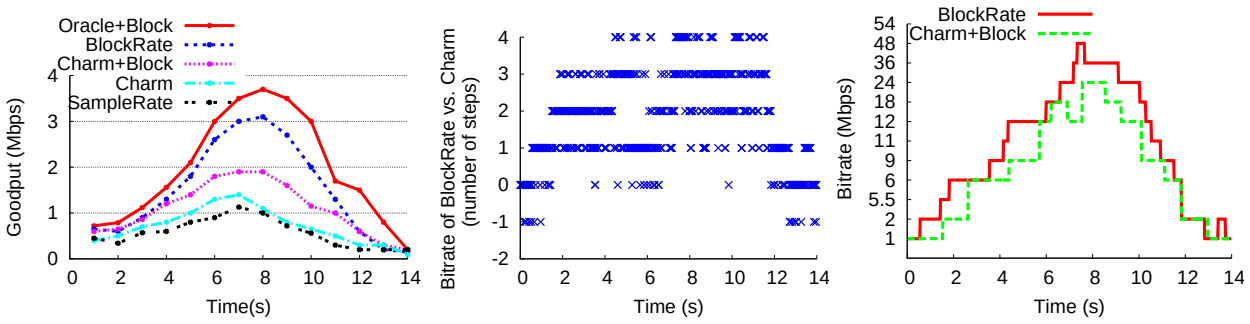


Figure 9: Vehicle-to-AP scenario.

4.2.2 *Vehicle-to-AP* [VEHICULAR2]

We use the VEHICULAR2 testbed for vehicle-to-AP experiments as the mobility or stationarity of the public transport vehicles in VEHICULAR1 is not under our control. Furthermore, as BlockRate requires sender as well as receiver modifications, it is not feasible to experiment with open access WiFi APs on the street. So we designate one laptop in VEHICULAR2 as a stationary AP in a downtown area with dense buildings, while the other is mounted in a car moving along a straight road passing close to the AP, as shown in Figure 9. The AP continuously sends UDP packes to the vehicle as long as they are within range. We repeat this experiment and collect data for each of the four algorithms separately.

Figure 10(a) shows the goodput over time for the different strategies in this vehicle-to-AP setting. We also include the result of Oracle+Block as described in §2. BlockRate has a peak goodput of 3.1Mbps, which is 1.6× better than Charm+Block and 2.8× better than



(a) Vehicle-to-AP goodput achieved by different schemes.

(b) BlockRate vs. Charm: Effect of overhead amortization plus anticipation using blocks.

(c) BlockRate vs. Charm+Block: Effect of anticipation alone.

Figure 10: Vehicle-to-AP scenario: BlockRate improves goodput by up to 1.6× over Charm+Block and 2.8× over SampleRate and Charm. BlockRate selects higher bitrates than both Charm and Charm+Block.

SampleRate and Charm. Note that BlockRate achieves the highest gains between the 5^{th} and 11^{th} seconds when it is in close proximity to the AP. This observation is consistent with the results in §3.1.2 showing that the SNR changes sharply when the vehicle is near the AP.

Figure 10(a) also shows that there is still room for improvement compared to Oracle+Block, e.g., the peak goodput of Oracle+Block is 3.7Mbps, which is 20% higher than BlockRate. This difference is due to inaccuracies in estimating SNR using the path loss model and the location information obtained from GPS devices.

Figures 10(b) and 10(c) show the bitrate selected by each scheme over time. For visual clarity, we compare BlockRate and Charm in Figure 10(b) and BlockRate and Charm+Block in Figure 10(c). The former shows the number of bitrate steps by which BlockRate exceeds Charm's bitrate, while the latter plots the actual bitrates selected by BlockRate and Charm+Block. We show the former as a difference in bitrate steps because, unlike Charm+Block that changes the bitrate gradually (while keeping it fixed for at least a block's duration), Charm changes its bitrate on a per-packet basis, so plotting actual bitrates in Figure 10(b) makes it cluttered and difficult to discern a trend. SampleRate's bitrates are close to Charm's and are therefore omitted.

Figure 10(b) shows that BlockRate selects higher bitrates than Charm most of the time, which is consistent with the results in §2.3. This difference in bitrate reflects the combined benefit of overhead amortization as well as anticipation using BlockRate. Figure 10(c) shows the benefit of anticipation alone as both Charm+Block and BlockRate being block-based protocols achieve similar overhead amortization benefits. The figure shows that Charm+Block is unable to predict sharp SNR changes under vehicular mobility and consistently selects lower bitrates than BlockRate.

4.3 Indoor static and pedestrian mobility

Although the case for anticipatory, block-based bitrate control is strongest in fast-changing conditions, for the sake of completeness, we also evaluate BlockRate in a slow-changing indoor setting. We experiment with (1) a walking scenario where one laptop is static and the other laptop is moved around randomly across different parts of the building, (2) a static scenario where nodes in the MESH testbed communicate with each other.

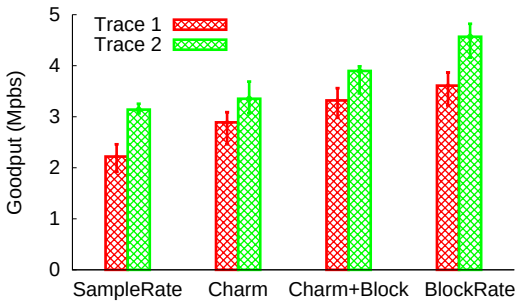


Figure 11: Pedestrian mobility: BlockRate achieves 1.4× higher goodput than Charm.

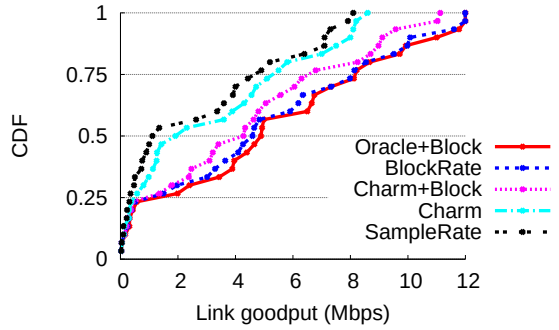
Pedestrian mobility.

Figure 11 shows the average goodput for the walking scenario for two sets of traces. The experiments were carried out at different locations of the computer science building. The error bars show the minimum and maximum values across 5 runs of each experiment. We observe that BlockRate performs 1.1× better than Charm+Block and nearly 1.4× better than Charm and SampleRate. This experiment shows that the linear regression model is useful even if the pedestrian mobility pattern is not strictly along a straight line, i.e., the speed or the rate of change of the relative distance between communicating nodes is not constant.

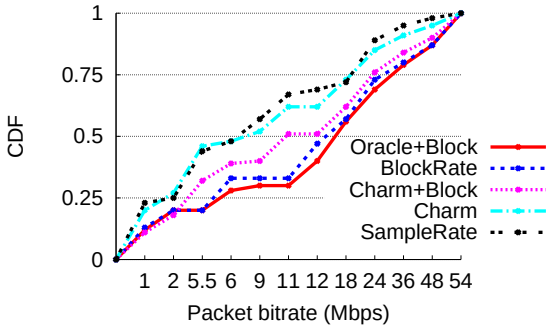
Static scenario.

We evaluate BlockRate in a static indoor setting over 30 links chosen randomly from the MESH testbed. We choose one link at a time and continuously send UDP packets for 100 seconds and measure the goodput.

Figure 12 shows the goodput and bitrate selection results for the different algorithms. Figure 12(a) shows the CDF of goodput across 30 links. BlockRate performs within 20% of Oracle+Block and improves median goodput by 4× over SampleRate and 2× over Charm. BlockRate yields little improvement over Charm+Block as there is little intra-block variation in channel quality. Figure 12(b) shows the CDF of the bitrates selected by these algorithms across packets. BlockRate selects fairly higher bitrate than both SampleRate and Charm.



(a) CDF of goodput across 30 links



(b) CDF of bitrate across all packets

Figure 12: CDF of goodput and bitrates on MESH: BlockRate improves median goodput by 2× over Charm.

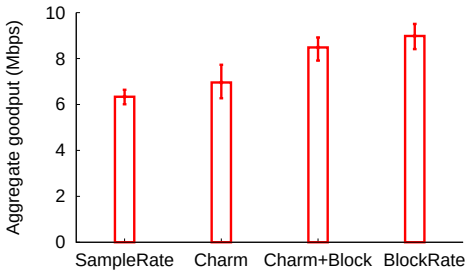


Figure 13: Goodput under interference in MESH: BlockRate achieves 1.3× higher goodput than Charm.

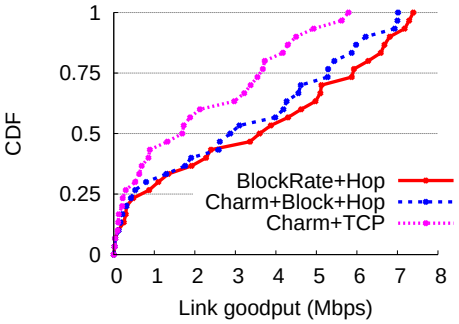


Figure 14: Reliable transfer in MESH: BlockRate+Hop improves median goodput by 2.1× over Charm+TCP.

4.4 Impact of interference

The indoor experiments described above were done in the presence of limited or no interference. Next, we evaluate the performance of BlockRate in an interference-dominated environment. This experiment conducted on the MESH testbed consists of two access points, each communicating with three clients. The nodes are selected such that they interfere with each other when transmitting concurrently. We measure the aggregate goodput across all nodes for each bitrate control scheme.

Figure 13 shows the aggregate goodput achieved by the different schemes with the error bars showing the minimum and maximum values across 5 runs. BlockRate improves goodput by 1.3× over Charm. As expected, SampleRate performs rather poorly, primarily because SampleRate cannot distinguish between interference and poor channel quality. The improvement over Charm+Block is small, which is unsurprising as this experiment involves a static scenario leaving little room for BlockRate's anticipatory control.

4.5 Reliable transfer performance

Although bitrate control schemes are commonly evaluated against the unreliable goodput metric, it is important to show that a proposed scheme performs well in conjunction with a reliable transport protocol. To this end, we next compare the reliable goodput achieved by block-based schemes in conjunction with a block-based transport protocol (Hop [12])) against that of packet-based schemes in conjunction with TCP. We do not present BlockRate in conjunction with TCP as its current implementation is split across the driver and user-space and interacts poorly with TCP's window control; we plan to fix this issue with a fully in-kernel implementation of BlockRate in the future. Unlike TCP, Hop uses a multi-round protocol to reliably transfer a block at each hop incurring low overhead, so it combines well with BlockRate. We conduct these experiments on the MESH testbed by randomly selecting 30 links. Each run of the experiment lasts for 100 seconds.

Figure 14 shows the CDF of the reliable goodput across links in the testbed for each bitrate control scheme. We observe that BlockRate+Hop outperforms Charm-Block+Hop and Charm+TCP by 1.2× and 2.1× respectively with respect to the median goodput.

Figure 15 evaluates the reliable goodput of different schemes under interference. The experimental setup is similar to §4.4 above. We observe that BlockRate outperforms Charm+Block by 10% and Charm by 22%. As above, the improvement over Charm+Block is small as this is a static scenario and BlockRate's anticipatory control is most effective in mobile scenarios.

4.6 Trace-driven evaluation

Next, we compare BlockRate against two non-SNR, packet-based schemes—SoftRate and RapidSample—that respectively rely on PHY and accelerometer-based movement hints. We use the indoor pedestrian mobility traces and ns-3 protocol implementations from [20] and [17] for this experiment and show the results in Figure 16. The box shows the mean goodput across 10 mobile traces and the error bar shows the minimum and maximum goodput. BlockRate outperforms packet-based algorithms SoftRate and RapidSample by nearly 1.3×. The increased throughput can be mainly attributed to the overhead amortization benefits of BlockRate.

A more in depth analysis shows that SoftRate spends approximately 56% of the time on per-packet backoffs and acknowledgements, while BlockRate only spends

Schemes	Monitored metrics	Mobility	Implementation/evaluation
SampleRate [13]	Transmit time	Static	802.11 testbed
RapidSample [17]	Loss rate, motion hint	Static, pedestrian	802.11/Click, Android, ns-3
RRAA [21]	Loss rate	Static, pedestrian	802.11testbed
MiRA [14]	Loss rate	Static, pedestrian	802.11n testbed
Charm [11], SGRA [22]	SNR	Static, pedestrian	802.11 testbed
BlockRate	SNR, distance	Static, pedestrian, vehicular	802.11 testbed
Holland et al. [10]	SNR	Slow- and fast-changing	ns-2 simulator
CK [6]	SNR	Static, vehicular	WARP
FARA [15]	SNR	Static	WiGLAN
SoftRate [20]	SoftPHY, BER	Static, pedestrian, fast-changing	USRP + ns-3 simulator
AccuRate [19]	Symbol dispersion	Static, pedestrian, fast-changing	USRP + channel simulator

Table 5: Comparison of BlockRate to recent bitrate control schemes. Above, *slow-* and *fast-changing* refer to simulated channels, while *pedestrian* and *vehicular* refer to mobile testbeds. Our evaluation shows that anticipatory block-based bitrate control outperforms representative packet-based schemes based on loss, SNR, and PHY or sensor hints.

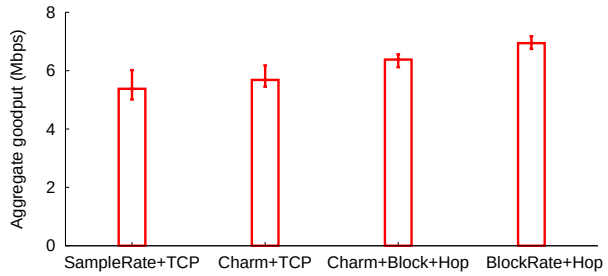


Figure 15: Reliable transfer in MESH: BlockRate+Hop outperforms Charm+TCP by 22% under interference.

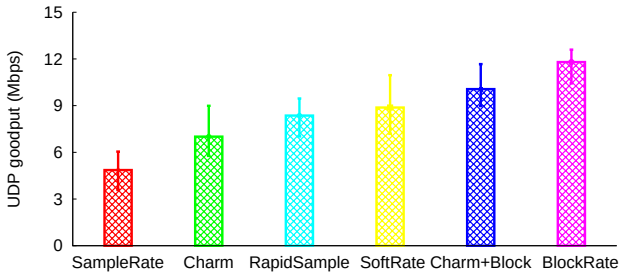


Figure 16: Trace-driven evaluation: BlockRate outperforms SoftRate and RapidSample.

35% of the time on these. A crude approximation would suggest that this reduction in overhead would translate to 1.5× goodput improvement. However, Figure 16 shows that BlockRate only has a 1.3× mean goodput improvement over SoftRate. An analysis of the loss patterns reveals that BlockRate has approximatley 15% higher loss than SoftRate because it provides a less accurate estimation of the SNR for the next packet. However BlockRate's lower responsiveness to channel conditions is more than offset by the overhead amortization gains resulting in higher goodput. An interesting question for future work is to determine if and to what extent BlockRate's design can be improved further using PHY information as opposed to its SNR-based approach.

5. RELATED WORK

BlockRate builds upon a large body of existing work in wireless bitrate control. What primarily distinguishes us is our goal, namely, to investigate the interaction of bitrate control with block transmission and to design a bitrate control algorithm optimized for blocks. Furthermore, we extensively evaluate BlockRate over a large-scale vehicular testbed involving naturally occurring mobility and environment patterns in addition to static or pedestrian mobility scenarios as in prior work.

To place BlockRate in relation to existing work as well as to justify the choice of compared schemes, we classify different bitrate control schemes in Table 5 along the following dimensions that we discuss in turn.

Monitored metric. Existing bitrate control schemes monitor a variety of metrics to estimate channel quality including the time to (successfully) transmit a packet [13], packet loss rate [21, 14], SNR [11, 15, 10, 18, 22], and more recently PHY-layer hints, e.g., SoftRate [20] uses "SoftPHY" or confidence values conveyed by PHY to learn the bit error rate (BER), and AccuRate [19] that improves upon this scheme by monitoring symbol dispersion and using it to jump to the best bitrate in a single step. In comparison, BlockRate uses a history-trained SNR-bitrate table similar in spirit to Charm [11] but with an important difference, namely, that BlockRate is designed to use mutiple bitrates predictively before it receives any new feedback about channel quality.

Blocks vs. packets. Using richer information about channel behavior such as PHY hints may further improve the performance of BlockRate, but is complementary to our primary goal of optimizing bitrate control for large blocks as opposed to small packets. PHY hints are particularly useful to react to ephemeral and fine-grained changes in channel quality, however this benefit comes at the cost of high per-packet feedback overhead. Our experiments in §2 and §4.6 suggest that there is more value in amortizing this overhead even if it comes at the cost of ignoring fine-grained channel fluctuations. As shown in Table 5, that most existing bitrate control schemes are implicitly designed for per-packet feedback. One exception is MiRA [14], a scheme that is evaluated over proprietary 802.11n cards using blocks, but their focus is on optimizing bitrate control in MIMO settings that is complementary to our goals.

Mobility. Although some prior works have experimented with pedestrian mobility [21, 14, 11] and others have experimented with vehicle-like mobility using simulated channel models [20, 19, 10] (marked *fast-changing* in Table 5) , we are not aware of a bitrate control scheme that has been evaluated under vehicular mobility with the sole exception of [6]. In comparison to Camp and Knightly [6] who experiment with small-

scale vehicle-to-AP scenarios, our vehicular mobility experiments additionally involve thousands of vehicle-to-vehicle contacts that occur naturally between public transit vehicles.

Implementation. BlockRate's design as well as the choice of alternate algorithms we were able to easily compare it against was dictated in part by our goal of being immediately deployable in widely used commodity wireless cards. This limited the algorithms we could compare against in real deployment to the first five rows in the table for which 802.11 implementations were available. Out of these, we picked one SNR-based scheme (Charm) and one non-SNR based scheme (SampleRate). SoftRate and RapidSample require PHY-layer hints or peripherals to sense movement, so we compared BlockRate with them through trace-driven simulation using the authors' implementation and traces. We could not compare against MiRA [14] as its implementation is based on a proprietary 802.11n card.

The basic idea of monitoring the SNR and learning the best bitrate for each SNR value is characteristic of most SNR-based schemes, and BlockRate is not particularly novel in this respect. Indeed, it is also susceptible to known limitations of SNR-based schemes (e.g., refer [9]). However, our primary contribution is not the specific bitrate control scheme based on SNR, but on cross-layer interactions between bitrate control and per-packet overhead. Our position is that a careful study of such interactions is valuable to influence the design of any bitrate control scheme, including those based on non-SNR metrics, PHY hints or sensor hints.

Our work has left open a number of open issues. Although we have focused on bulk transfer goodput, the use of large blocks in BlockRate can adversely affect delay-sensitive applications like VOIP and live video streaming. For such applications, using a packet-based scheme and prioritizing their packets over those of block-based schemes can alleviate the problem (as also shown in prior work [12]). Another challenge is to make BlockRate work underneath TCP, as described in §4.5. Adapting BlockRate to leverage the MIMO features of 802.11n in conjunction with its native support for block transfer is also an open question. We plan to address these issues as well as that of reducing BlockRate's reliance on GPS (§3.1.2) as part of future work.

6. CONCLUSIONS

Recent trends in research as well as in practice (e.g., commodity 802.11n cards) suggest significant performance benefits of amortizing overhead across large blocks compared to small packets. However, state-of-the-art bitrate control algorithms are primarily designed to react on a per-packet basis, so they are forced to trade-off the gains of amortizing overhead using blocks against the performance loss due to the reduced responsiveness of bitrate control.

Our contribution, the design and implementation of BlockRate, shows that the benefits of blocks can be leveraged without compromising on the responsiveness of bitrate control. The key insight in BlockRate is to convert the hurdle, namely large feedback delay with blocks, to an opportunity by predicting coarse-grained channel quality trends over the course of block transmission delays. Our measurements under a variety of typically encountered vehicular and pedestrian mobility scenarios show both that it is possible to predict channel quality trends at coarse time scales and that leveraging these predictions results in significant gains in unreliable as well as reliable goodput.

Acknowledgments: We gratefully acknowledge the feedback received from our shepherd James Roberts and the anonymous reviewers. This work was supported in part by funding from the NSF grants CNS-0845855, CNS-1040781, and CNS-0910671.

7. REFERENCES

[1] Linear regression. http://en.wikipedia.org/wiki/Linear_regression.
[2] Madwifi. http://madwifi-project.org.
[3] ns3 simulator. http://www.nsnam.org.
[4] Proceedings of the IEEE: Special Issue on Adaptive Modulation and Transmission in Wireless Systems, 2007.
[5] 802.11n: The next generation of wireless performance. Cisco White Paper, 2009.
[6] J. Camp and E. Knightly. Modulation rate adaptation in urban vehicular environments: Cross-layer implementation and experimental evaluation. In *MobiCom*, 2008.
[7] A. Duel-Hallen. Fading channel prediction for mobile radio adaptive transmission systems. Proceedings of the IEEE, Vol. 95: pp. 2299-2313, 2007.
[8] A. Goldsmith and S. Chua. Adaptive coded modulation for fading channels. IEEE Transactions on Communication, Vol. 43: pp. 595-602, May 1998.
[9] D. Halperin, W. Hu, A. Sheth, and D. Wetherall. Predictable 802.11 packet delivery from wireless channel measurements. SIGCOMM, 2010.
[10] G. Holland, N. Vaidya, and P. Bahl. A rate-adaptive mac protocol for multi-hop wireless networks. MobiCom, 2001.
[11] G. Judd, X. Wang, and P. Steenkiste. Efficient channel-aware rate adaptation in dynamic environments. MobiSys, 2008.
[12] M. Li, D. Agrawal, D. Ganesan, and A. Venkataramani. Block-switched networks: a new paradigm for wireless transport. In *NSDI*, 2009.
[13] R. T. Morris, J. C. Bicket, and J. C. Bicket. Bit-rate selection in wireless networks. Technical report, Masters thesis, MIT, 2005.
[14] I. Pefkianakis, Y. Hu, S. H. Wong, H. Yang, and S. Lu. Mimo rate adaptation in 802.11n wireless networks. MobiCom, 2010.
[15] H. Rahul, F. Edalat, D. Katabi, and C. Sodini. Frequency-Aware Rate Adaptation and MAC Protocols. In *MOBICOM*, 2009.
[16] T. Rappaport. *Wireless Communications: Principles and Practice*. 2nd edition, 2001.
[17] L. Ravindranath, C. Newport, H. Balakrishnan, and S. Madden. Improving wireless network performance using sensor hints. In *NSDI*, 2011.
[18] B. Sadeghi, V. Kanodia, A. Sabharwal, and E. Knightly. Opportunistic media access for multirate ad hoc networks. MobiCom, 2002.
[19] S. Sen, N. Santhapuri, R. R. Choudhury, and S. Nelakuditi. Accurate: constellation based rate estimation in wireless networks. In *NSDI*, 2010.
[20] M. Vutukuru, H. Balakrishnan, and K. Jamieson. Cross-Layer Wireless Bit Rate Adaptation. In *ACM SIGCOMM*, 2009.
[21] S. Wong, H. Yang, S. Lu, and V. Bharghavan. Robust rate adaptation for 802.11 wireless networks. In *Mobicom*, 2006.
[22] J. Zhang, K. Tan, J. Zhao, H. Wu, and Y. Zhang. A practical snr-guided rate adaptation. Infocom, 2008.