In this section, simulation results of the compression ratio are used to validate the analysis in the previous section. Further validation is provided with an implementation of the delta compression scheme for a content downloading application in an 802.11b wireless network.

A. Simulation Results for Compression Ratio

A Matlab simulation is carried out to verify the compression ratio analysis. Packet $X_k$ and $V$ are generated as two i.i.d. random sequences that follow a discrete uniform distribution between 0 and 255 with a packet length of 1,500 bytes. Packet $X_{k+1}$ is generated according to the simplified content generation model. Considering $X_k$ and $X_{k+1}$ as two byte strings, our lossless delta compression algorithm is designed as follows. Between $X_{k+1}$ and $X_k$, $X_{k+1}$ is regarded as a string with many substrings copied from $X_k$ and new bytes. Each substring is coded by its length and location in $X_k$. No assumptions about the format or alignment of the input data are made. The flowchart for our compression scheme is depicted in Figure 1. Essentially, this can be broken down into two steps: decomposing $X_{k+1}$ into substrings copied from $X_k$, coding the length and location of each substring $i$ (len$_i$, position$_i$) by a number of bits. In Step 1, obtaining the optimal partition for $X_{k+1}$ is a challenging topic. Several methods have been proposed to increase the searching speed and efficiency. For example, a hash function can be used for searching the longest matched substring. However, for short packets (say less than 1,500 bytes) or non-real time compression, hash tables are not necessary. One can simply search the current byte $X_{k+1}(i)$ in $X_k$ directly and look for the longest match. In Step 2, because len$_i$ cannot be guaranteed to be larger than 1 (otherwise a matched substring cannot be found), a modified substring coding scheme is used by adding an index bit: a single byte is coded by 9 bits ($L_1 = 1.125$ bytes) with 1 bit for index and 8 bits for data content; a matched substring is coded by 14 bits ($L_2 = 1.75$ bytes) with 1 index bit, 8 bits for length, and 5 bits for location.

In Figure 2, the compression ratio for our lossless delta compression algorithm as well as the entropy curves are plotted for different $p$, which represents the degree of similarity between $X_{k+1}$ and $X_k$. From Figure 4, it is clear that the delta compression scheme can only be applied for larger values of $p$ between 0.45 and 1 (i.e., 55% of the time). Intuitively, this makes sense since larger compression gains are obtained when there is large similarity between adjacent packets. Additional coding methods (e.g., Huffman coding) can be used to reduce these overheads associated with fixed length coding.

B. Implementation

We implement, using C++, a fast content downloading application based on the delta compression scheme. The scheme gives an excellent compression gain (and yields a short delta) but not necessarily the best delta creation time. Three bytes are used to code each substring. We employ a set of HTML Web pages with different content and random lengths from a forum website, and save them at a content server in both Delta compressed and uncompressed formats. The client downloads and decompresses the web pages one after the other in real time over a 1 Mbps 802.11b wireless link. Results in figure 3 show that the average real-time throughput improvement using delta compression is about 4 times more than the case when there is no compression. This takes into account the overheads needed to transmit and decode the delta file. The average static compression gain is also more than twice that of the commercial WinZip™ software.
Searching for $X_i(\text{position}, \text{len})$ with the longest $\text{len}$, s.t. $X_i(\text{position}, \text{len}) = X_{i+1}(\text{position}, \text{len})$

$\text{len} \geq 1$

Code $X_i(\text{position}, \text{len})$ as a copied substring of $X_{i+1}$

$i = i + \text{len}$

Code $X_{i+1}(0)$ itself as a new byte of $X_{i+1}$

$i = i + 1$

Figure 1: Flowchart for Substring Search

Figure 2: Compression Ratio of Delta Algorithm
<table>
<thead>
<tr>
<th></th>
<th>Uncompressed</th>
<th>Delta compressed</th>
<th>WinZip compressed</th>
</tr>
</thead>
<tbody>
<tr>
<td>File size</td>
<td>44 KB</td>
<td>3 KB</td>
<td>7 KB</td>
</tr>
<tr>
<td>Equivalent Throughput</td>
<td>303 Kbps</td>
<td>1,171 Kbps</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Compression ratio, \( \eta = \frac{\text{Compressed size}}{\text{Uncompressed size}} \)

Average throughput, \( T = \frac{\text{Transmitted bits}}{\text{Time interval}} \)

Equivalent throughput = \( T \eta \)

Figure 3: Delta Compression Performance