Delayed-Dictionary Compression for Packet Networks

Yossi Matias
Tel Aviv University

Raanan Refua
Tel Aviv University

We consider data compression in packet networks, in which data is transmitted by partitioning it into packets. Packet compression allows better bandwidth utilization of a communication line resulting in much smaller amounts of packet drops, more simultaneous sessions, and a smooth and fast behavior of applications.

Packet compression can be obtained by a combination of *Header compression* and *payload compression*, which are complementary methods. In this work we focus on payload compression only. We are particularly interested in dictionary-based compression. Many dictionary compression algorithms were developed, following the seminal papers of Lempel and Ziv.

In dictionary compression, an input sequence is encoded based on a dictionary that is constructed dynamically according to the given text. The compression is done in a streaming fashion, enabling to leverage on redundancy in the input sequence.

In many packet networks, including ATM, Frame Relay, Wireless, and others, packets are sent via different routes, and may arrive reordered, due to different network characteristics, or due to retransmissions in case of dropped packets. Since streaming compression assumes that the compressed sequence arrives at the decoder at the order in which it was sent by the encoder, the decoder must hold packets in a buffer until all preceding packets arrive. This causes *decoding latency*, which may be unacceptable in some applications.

To alleviate decoding latency, standard packet compression techniques are based on a packet-by-packet compression. For each packet, its payload is compressed using a dictionary compression algorithm, independently to other packets. While the decoding latency is addressed properly, this may often result with poor compression quality, since the inherent redundancy within a packet is significantly smaller than the redundancy over many packets in the stream.

We introduce a novel compression algorithm suitable for packet networks: the *delayed-dictionary compression* (DDC). The DDC is a general framework that applies to any dictionary algorithm; it considers the dictionary construction and the dictionary-based parsing of the input text as separate processes, and it imposes a delay Δ in the dictionary construction. As a result, when decoding a packet, the decoder does not depend on any of the preceding Δ packets, eliminating or diminishing the problems of out-of-order packets and packet drops compared to streaming compression, still with a good traffic compression ratio.

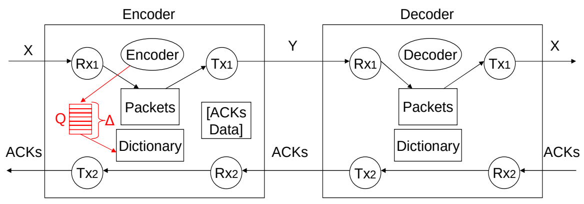

Figure 1: Internal structure of the Encoder and the Decoder: The encoder task transfers phrases to the dictionary task by using a FIFO queue.

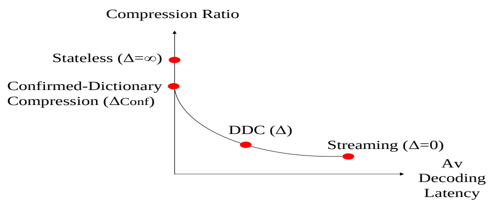

Figure 2: A tradeoff between the compression ratio and the average decoding latency. Streaming has the best compression ratio and the worst decoding latency. DDC has compression ratio close to that of streaming compression, and also has average decoding latency which is close to that of stateless. The *confirmed-dictionary compression* algorithm ensures a zero decoding latency.

The internal architecture of the encoder and the decoder combined with the DDC method is depicted in Fig. 1. The dictionary delay is implemented by a FIFO queue which is initialized to Δ dummy packets.

We focus on two alternative encoding methods for the DDC algorithm. The first adapts to the network propagation delay and the probability for packet loss. The second, called *confirmed-dictionary compression*, ensures zero decoding latency. The DDC at its most powerful version ensures that the compression ratio will be at least as good as that of stateless compression, and quite close to that of the streaming compression, with decoding latency close to or equal to that of stateless compression.

A full tradeoff between compression ratio and decoding latency can be obtained using the DDC algorithms, as illustrated in Fig. 2. On one extreme of the tradeoff is the streaming compression, which is DDC with $\Delta = 0$, it has the best compression ratio and the worst decoding latency. On the other extreme we have the DDC with confirmed dictionary, which has zero decoding latency - as in stateless compression; its compression ratio is the worst among the DDC algorithms, but is still better than that of stateless compression. Thus, the DDC has the benefits of both stateless compression and streaming compression.

With the right choices of the dictionary delay parameter, it may have a decoding latency which is close to that of stateless compression, and with compression ratio which is close to that of streaming compression. For example, for a concatenation of the Calgary corpus files, fragmented into packets with a payload of 125 bytes, in streaming the compression ratio is 0.52 with an average decoding latency of 62 packets, while in DDC with a large dictionary delay of 200 packets we obtain a compression ratio of 0.68 and only an average decoding latency of 14.3 packets.

The DDC method is particularly good for low to medium speed communication links. Its advantage is most significant for applications in which the latency is important, and in which the order of decoded packets is not important.

We conducted various experiments to establish the potential benefit of DDC, by comparing the compression ratios of streaming compression versus stateless compression, and

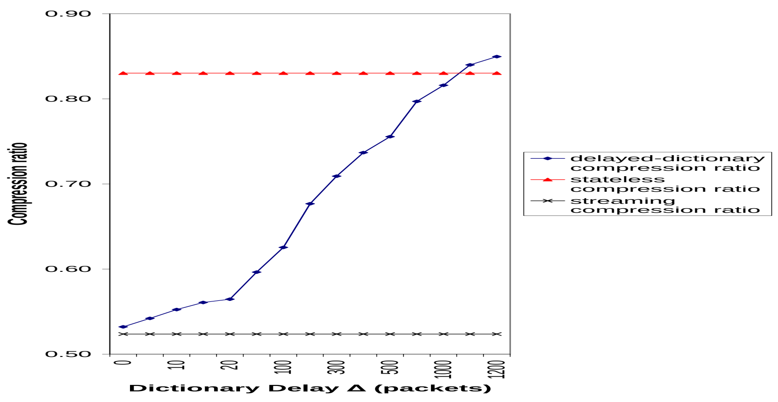

Figure 3: Compression ratio of DDC as a function of the dictionary delay in packets, compared to stateless compression ratio and to streaming compression ratio. The ratio for streaming compression is very close to the DDC ratio with zero dictionary delay. The data file in use is the concatenation of 18 Calgary corpus files, $|Header| = 20, |Payload| = 125$. DDC obtains a good compression ratio even for large dictionary delays.

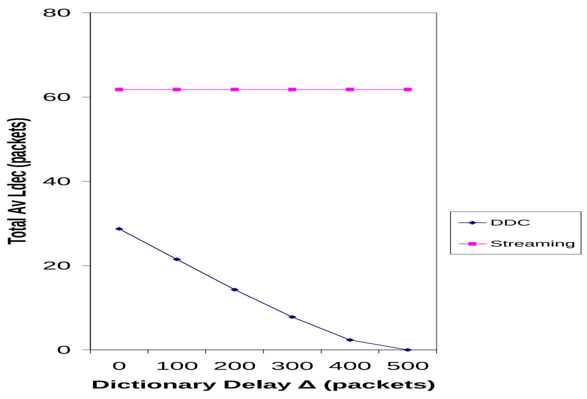

Figure 4: The Effect of the Dictionary Delay on the Decoding Latency: Increasing the dictionary delay will cause a decrease of the decoding latency L_{dec}. As can be seen, the average decoding latency in DDC is smaller than that of streaming. In particular when $\Delta = 500$ packets in DDC do not wait at all while packets in streaming wait on average for 62 packets. $RTT = 5000$msec.

by measuring the decoding latency of streaming compression. We used the Flexible Parsing version of LZW as the compression algorithm for testing purposes. For network related experiments we used the Planet Lab project over the Internet.

Using experimentation, we study the dependency of compression quality and the imposed dictionary delay, showing that the improvement in compression over stateless compression could be significant even for a relatively large dictionary delay. The compression ratios of stateless compression, streaming compression, and DDC, for small packets, are depicted in Fig. 3.

We also consider the effect of the dictionary delay on the performance in terms of the decoding latency. This effect is depicted in Fig. 4. A sufficiently large dictionary delay will practically provide a practical zero decoding latency. We compare the decoding latencies of streaming compression vs. DDC, showing that the latter is indeed considerably better. For streaming compression, the maximal decoding latency is $4RTT$ (e.g., for $RTT = 5000$msec and a payload size of 125 bytes the decoding latency is 963 packets) while in DDC we can control it to be zero.