# Abstract

A digital twin is a virtual representation of a system that is updated from real-time data. A digital twin can provide accurate information about the current state of the system, and allow for predictions of future states of the system.

Motorway traffic has been growing year on year with the ever increasing people and vehicle population. With this growth comes the risk of increased congestion, traffic accidents, and roadworks. These issues could be reduced with real-time traffic monitoring. A method of real-time traffic monitoring could be possible with a digital twin of the motorway. In order to create a digital twin, traffic data from the motorway is needed. This data must be real time and readily available. Several sensors exist on modern motorways such as inductive loops, road gantry cameras, and radar detectors. Others data sources can be found directly on vehicles such as GPS devices. The data from these sensors can be used to feed the digital twin.

Apache Kafka is a distributed messaging service that is based on a publisher-subscriber architecture. In Apache Kafka, data can be partitioned into 'partitions' that are within different 'topics'. This research explores the use of Apache Kafka as the communication service between the motorway sensors and the digital twin. The objective is to create a highly available, high throughput, and low latency design. The proposed system implements topics per sensor group and partitioning per sensor. The research also explores the properties of the motorway sensors, focusing on their latency and the type of data they produce.

The proposed Apache Kafka data delivery architecture is evaluated using simulated data streams. These simulated data streams are taken from virtual sensors which draw data from a SUMO simulation of the M50 motorway in Dublin. Several simulations are conducted using various Kafka configurations. The effects of compression, batching, and varying the broker hardware are tested, as well as varying the number of brokers and consumers. The effects of the relative distance between the clients and Kafka are also tested.

It was found that single broker designs offer the highest performance in terms of latency, however these systems have a single point of failure. It was found that compression can decrease the latency by approximately 7%. Compression also increases the throughput of the system by 40%. The use of multiple consumers also improves the throughput of the system by 21%. Reducing the distance between Kafka and its clients reduced the latency of the system, however throughput is unchanged.

The final solution is a 3 broker configuration. This system uses compression to decrease the latency, and replication across the brokers to increase the availability of the service. The final solution has a dedicated consumer per topic. This system experiences 145 ms of latency in low traffic simulations, and 156 ms latency in high traffic simulations with ideal cluster placement.