# Homework 3 - Bonus

The paper presents a new approach for efficient and reliable processing of streaming data using Spark. It concentrates on the concept of DStreams (Discretized Streams) referring to the way Spark works with real-time data streams in small batches or micro-batches. This enables the processing of continuous data in such a way that, in case of failure, no data would be lost, means in a fault-tolerant manner.

From this paper, I understood that DStreams represent data streams as a series of small batches, processed over time. These small batches allow Spark to treat streaming data in the same way as batch data, using its existing operations like map, flatMap, reduce, and filter. This makes it easier to handle large amounts of real-time data, such as stock market data, online shopping activity, health monitoring or social media feeds, because the data is processed in manageable chunks.

The second key aspect is that DStreams are built on top of a Resilient Distributed Dataset, which is the foundation for Spark's fault-tolerant design. If any failure happens in the process, then Spark will recover the lost data by recomputing from other parts of the system because of storing multiple duplicate files in the cluster node. As DStreams have this characteristic to recover the data without the loss of any streaming input, therefore they are much more fault tolerant and reliable to use.

This paper also discusses how DStreams may distribute work across multiple workstations in a cluster, making it easier to manage large amounts of data. Even if the volume of the data increases, the Spark can continuously process data quickly and effectively. Spark can scale up and handle large data without any lag. Because of this Dstreams a great solution for real-time processing in big or large systems.

Limitation in this paper: One of the major drawbacks of DStreams is that it only process data in small batches, which results in small delay between data generation and processing. Such latency can be an issue for applications that demand immediate processing, such as livestock trading even a little latency might make the biggest difference. This makes DStreams not always ideal in more complex event processing cases, which involve fast decision-making. And also, that batch processing the data may make it insufficiently fast for applications that demand real-time continuous analysis.

Conclusion: This paper describes how DStreams represent a scalable and fault-tolerant way to process streaming data using Spark. Even though it does very well in many real-time applications, the micro or small batch approach can introduce latency in processing the data, hence it is less suitable for applications that require fast or immediate processing like Stock Market applications, Health monitoring applications etc. Though they may not be the best option for applications that require low latency, DStreams are nonetheless a good option for managing massive amounts of real-time data.