MONASH University

MONASH
INFORMATION
TECHNOLOGY

FIT5202 – Data Processing for Big Data

Stream Data Processing

Prajwol Sangat, Ting Chee Ming

Updated by Jay Zhao (Sep/2024)

GROUP OF EIGHT AUSTRALIA

Last 8 Topics

- Parallel data processing
- Machine learning algorithms

This Week

- Stream data processing
- Streaming processing technology

Data Stream Applications

#trending

Track monitoring

#Stock market

#Security

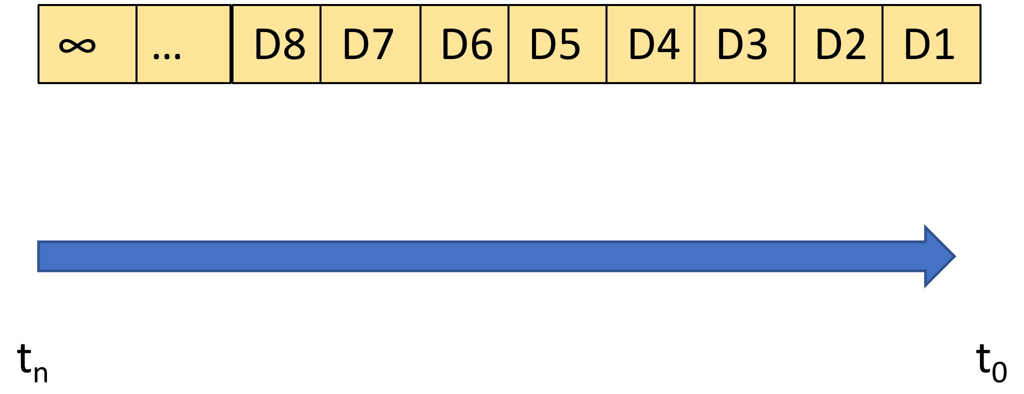
Characteristics of the application

- Unbounded (infinite) data – data keeps coming in without boundary
- Real time query at any given point in time
  - Get response quickly
  - Differ from batch query applied to entire table/data block
- Data comes continuously
  - Uniform rate (e.g., data comes every 1 second)
  - Bursty (e.g., data records come in one single reading)
- Interested in an “event” or trends across time.

Data stream

A data stream is a real-time, continuous, time-ordered (implicitly by arrival time or explicitly by event timestamp) sequence of items.

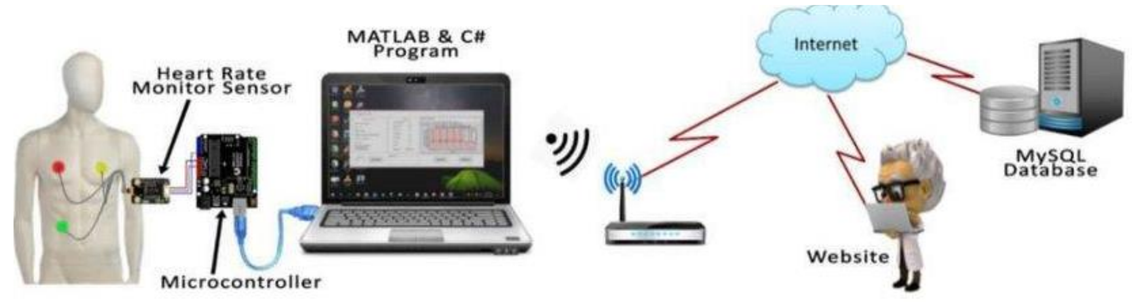
#Stock market

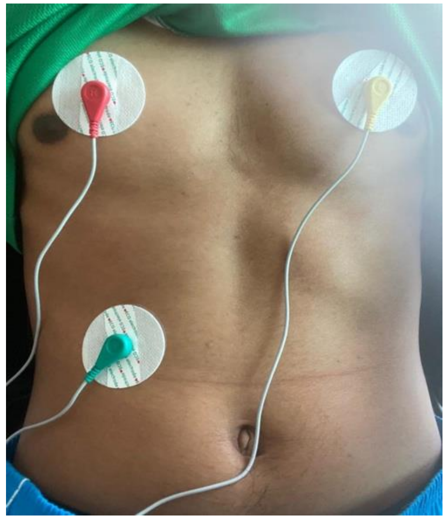


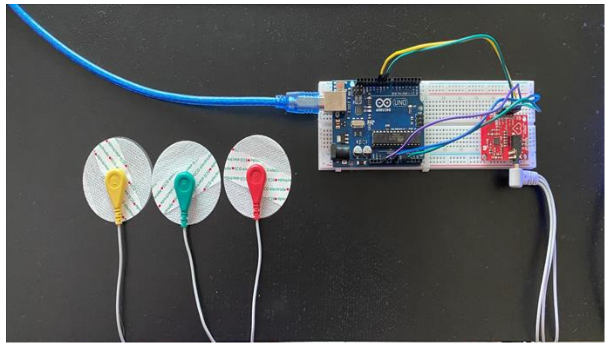
Arrival time: The time data arrives at processing end
Event timestamp: The actual time data is generated

Cardiac Monitoring with ECG

Online monitoring of heart conditions to detect any abnormal heart beats





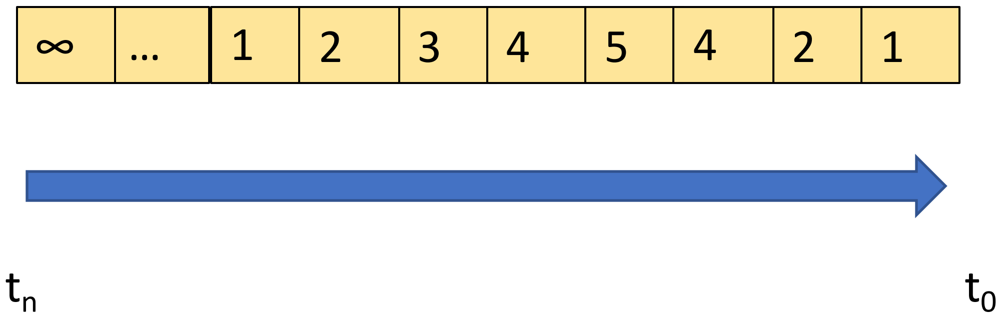


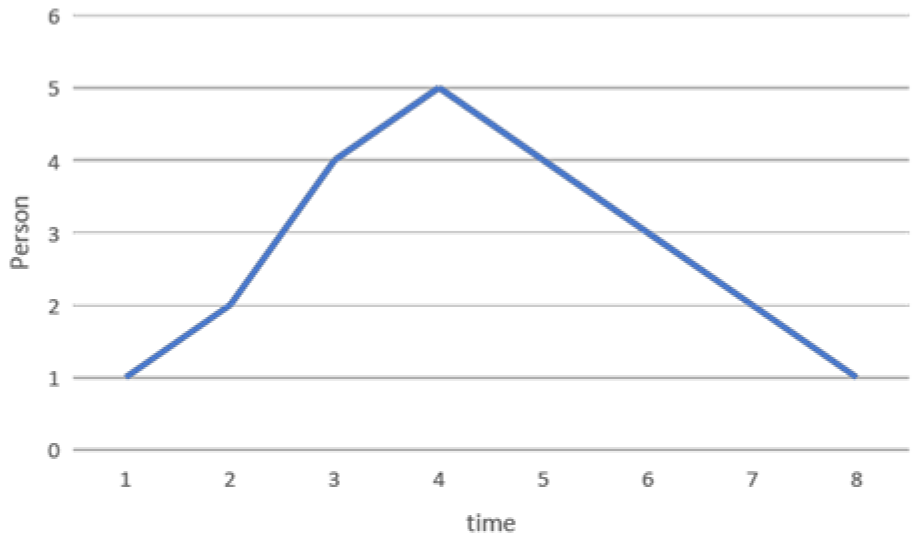
Data Stream Query - Examples

What is the total number of attendees up to tx?

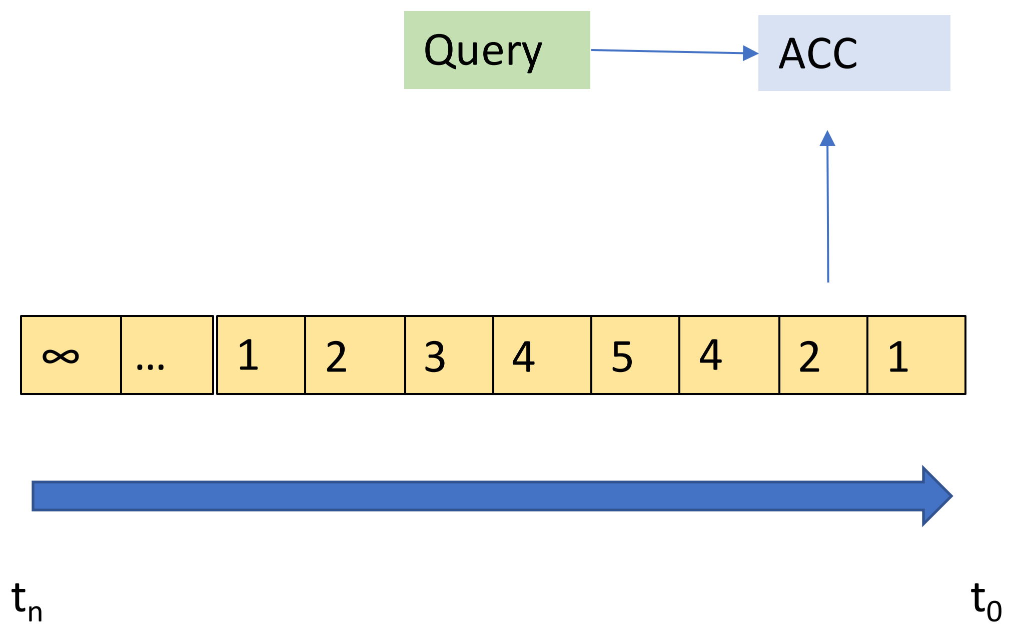
What is the average number of attendees so far? (at time tx)

Will the next number be lower or higher than the current reading?





What is the total number of attendees up to tx?



Assume memory can only keep 5 tuples.

Keep an accumulator for the sum. There is no need to keep the tuples.

Possible issues:

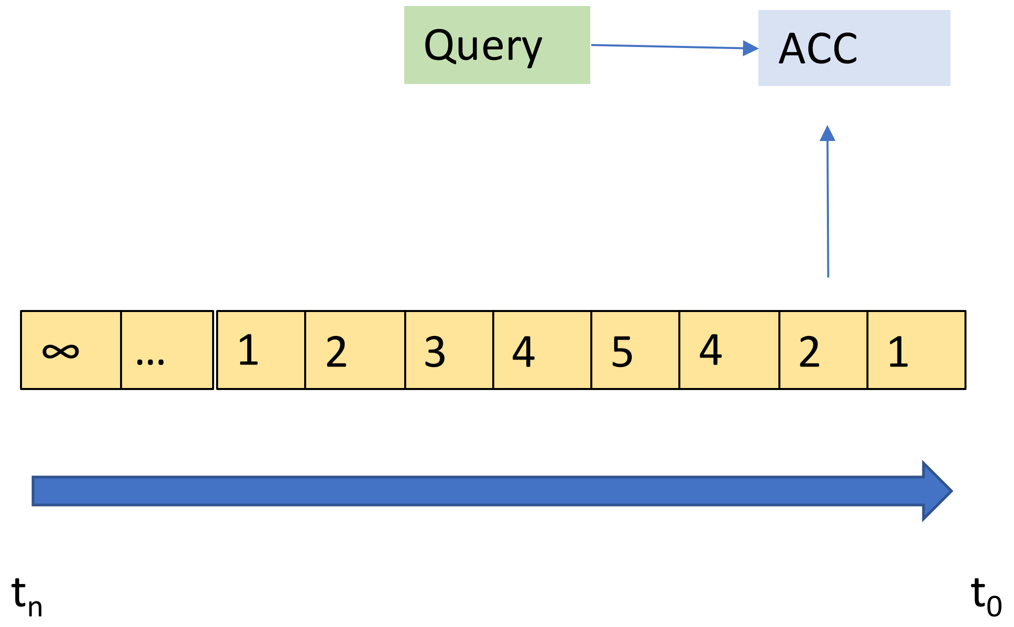
- What to do with incoming tuples if the incoming tuple arrival rate > processing time?

https://aseigneurin.github.io/2018/08/27/kafka-streams-processing-late-events.html

Data streams: Can't keep all incoming data, because data is unbounded
Database: Have persistent storage to store all data

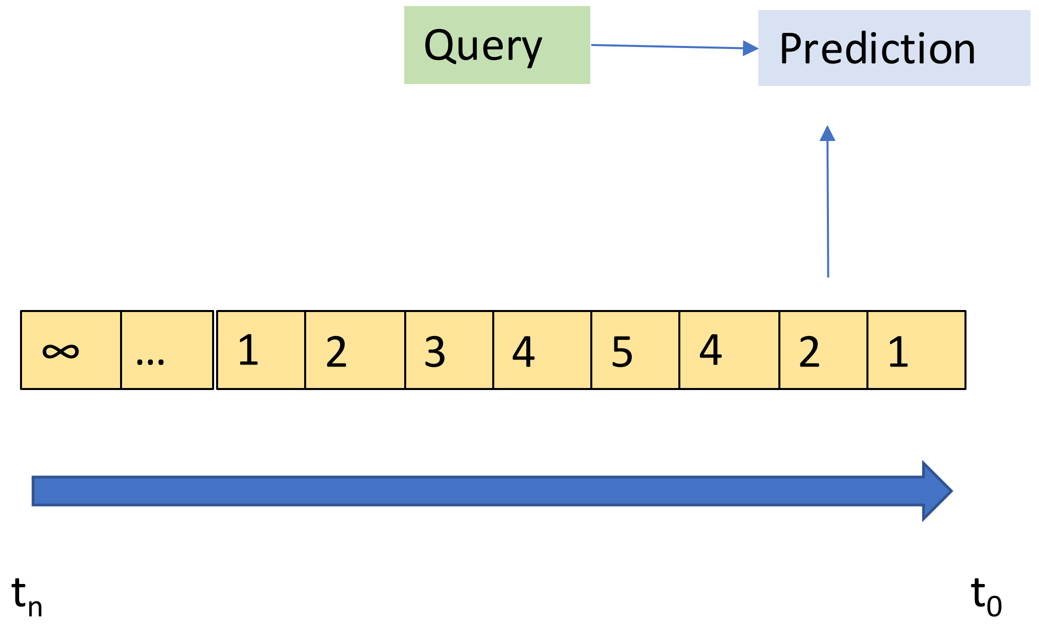
→ Rely on accumulator as a summary of the stream

What is the average number of attendees so far? (at time tx)



Keep two accumulators for the sum and count. There is no need to keep the tuples.

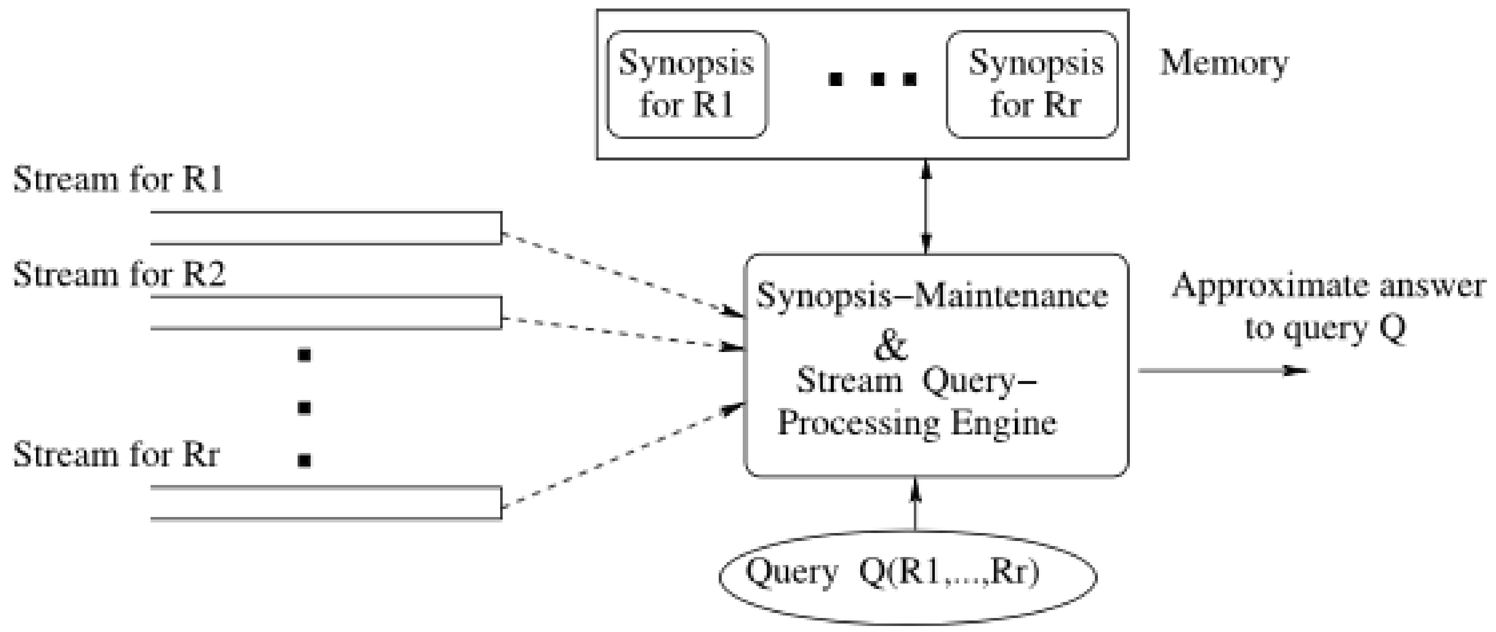
Will the next number be lower or higher than the current reading?



Prediction may need all/some previous tuples.
Memory can only keep 5 tuples.
What can we do?

Learn from previous data (create summary/model based on previous data)

Stream Processing System Architecture



Garofalakis M., Gehrke J., Rastogi R. (2016) Data Stream Management: A Brave New World. In: Garofalakis M., Gehrke J., Rastogi R. (eds) Data Stream Management. Data-Centric Systems and Applications. Springer, Berlin, Heidelberg

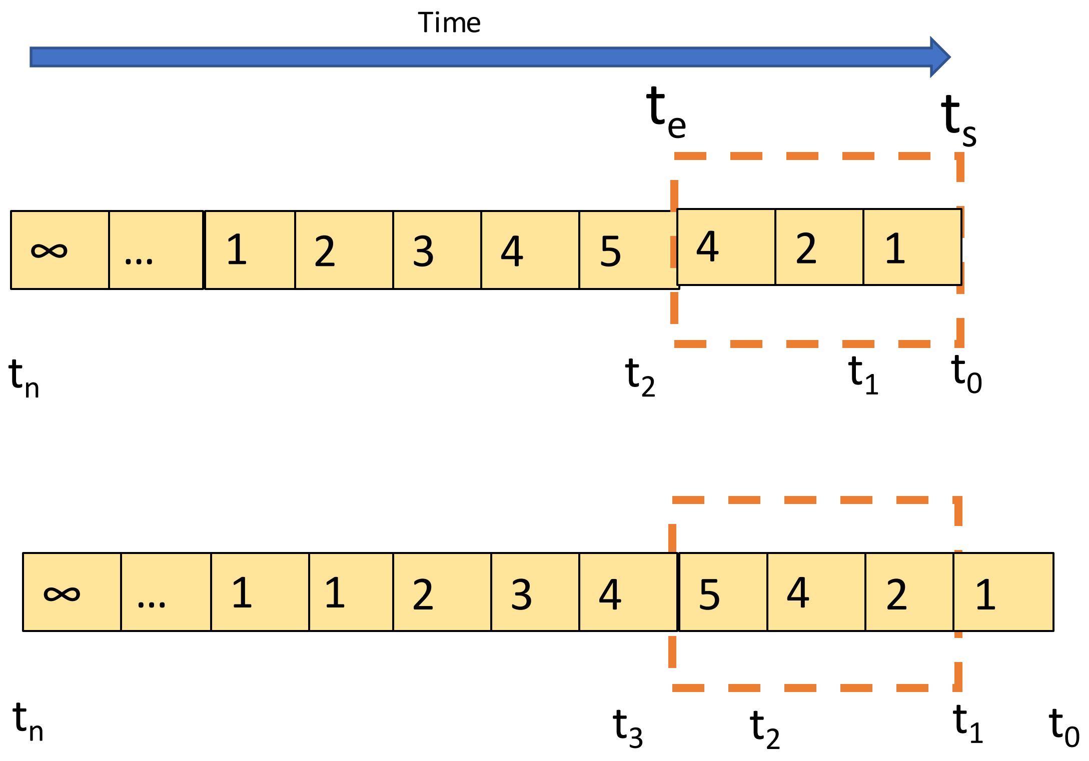
Time Decay

- Is it possible to reduce the importance of older tuples, without eliminating their influence on the results of the analysis?
  - Some older tuples will not be as useful as newer tuples

- Popular approach: windowing (consider recent past data in window)
  - Query processing is performed only on the tuples inside the window.
  - Divide unbounded data into some bounded set using window mechanism

- Window types:
  - **Time-based**
  - **Tuple-based (count-based)**

Stream Window – Time Based Window



Example:

t_2-t_0 $= 2\ seconds$

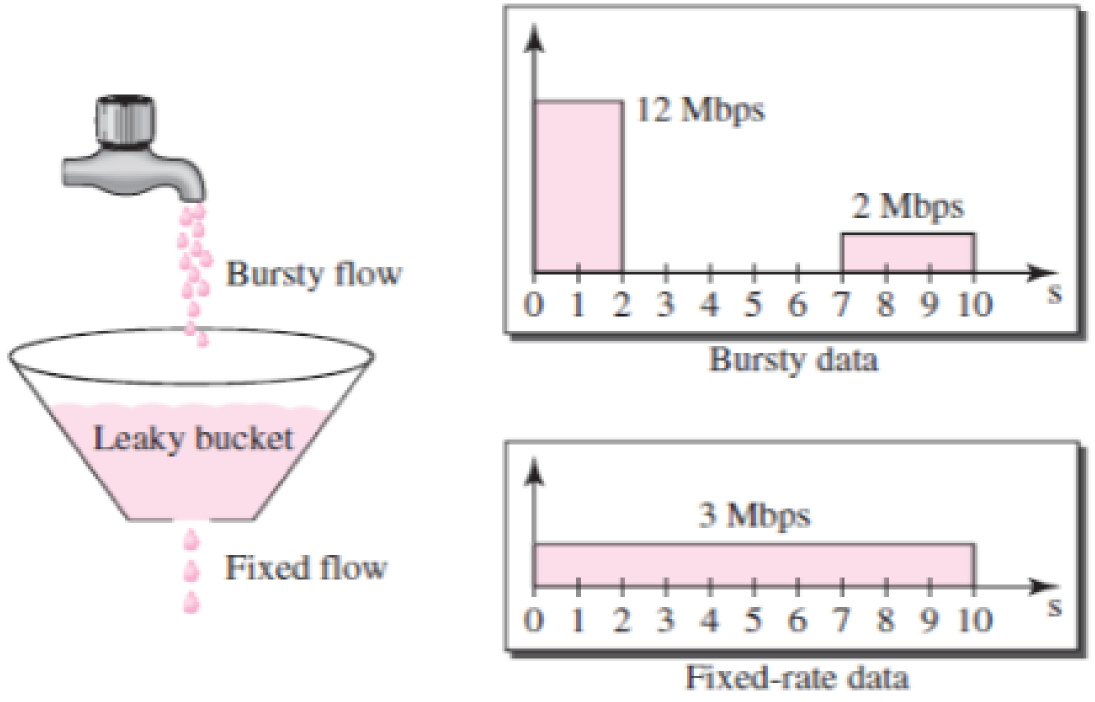
t_3-t_1 $= 2\ seconds$

Sliding windows

Window size 2 seconds
Slides 1 second.

- The size of a window is specified by a fixed time duration
- The number of tuples in windows may vary depending on how many tuples are available while processing
- It may be high due to the bursty arrival of data due to delay in data transfer
- Tuples in the windows will be processed, no matter how many tuples are in the windows

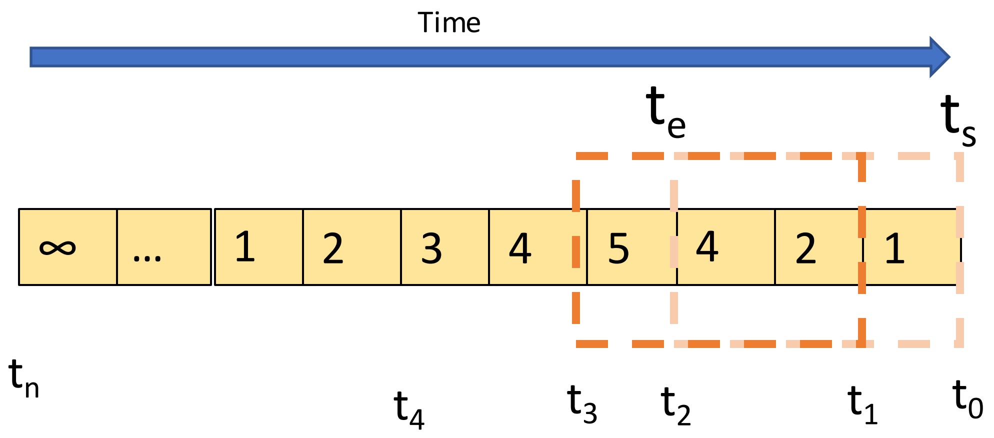
Bursty data



Data rate change suddenly in short time

Image source: https://www.quora.com/What-is-the-difference-between-token-bucket-and-leaky-bucket-algorithms

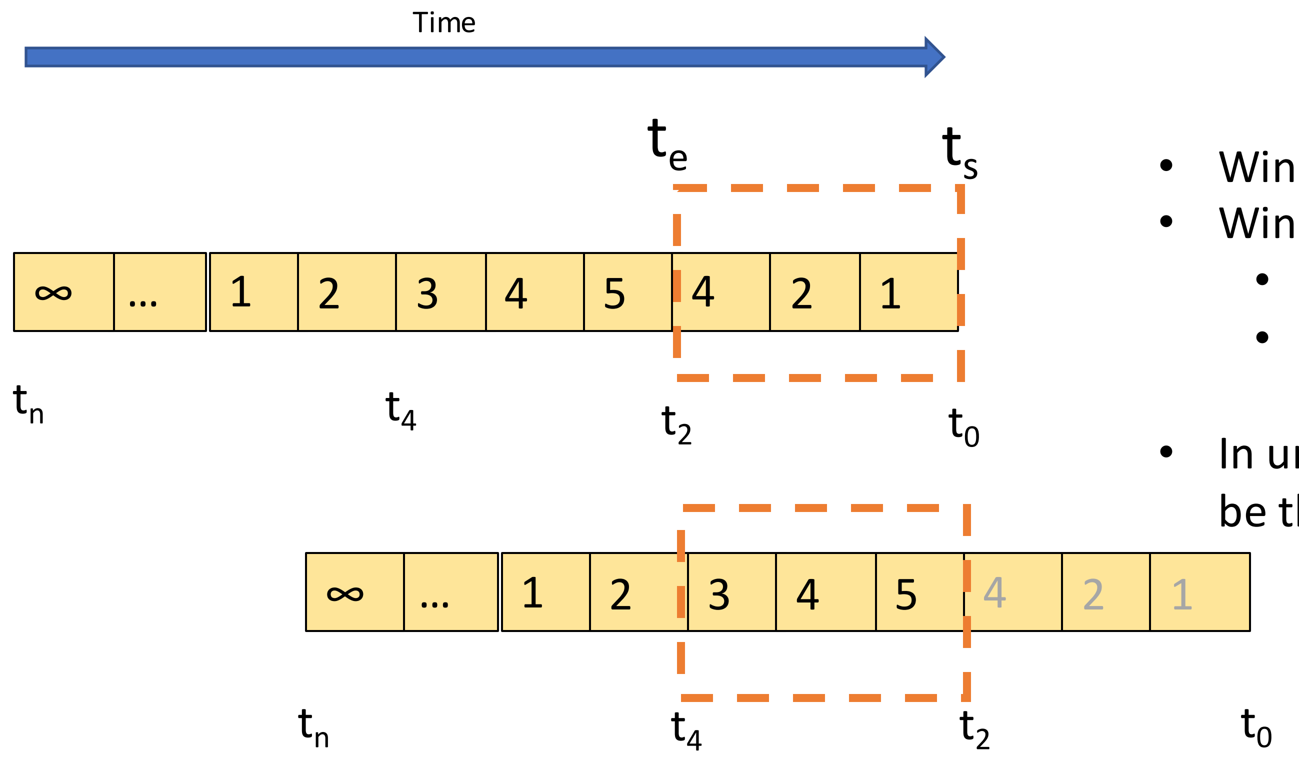
Stream Window – Time Based Window



Overlapping Sliding Window

Window size 2 seconds
Slides 1 second.

Stream Window – Time Based Window

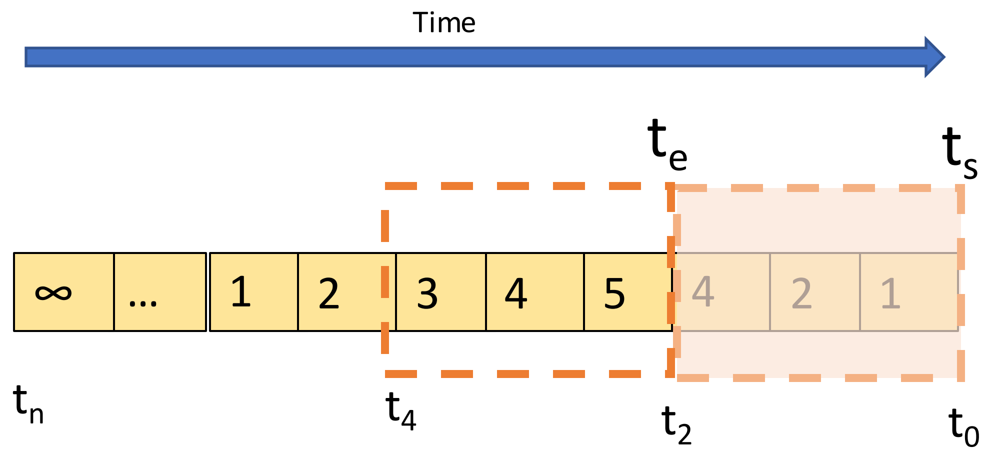


- Window size is based on time, eg 2 seconds
- Window can be advanced/slid by:
  - t_e-t_s (window size)
  - Duration less than the window size (sliding window).
- In uniform data rate, the number of tuples will be the same for each window.

Non-Overlapping Sliding Window

- Slide of window is equal to size of window
- Two windows are disjoint
- No influence of data in previous window on current window

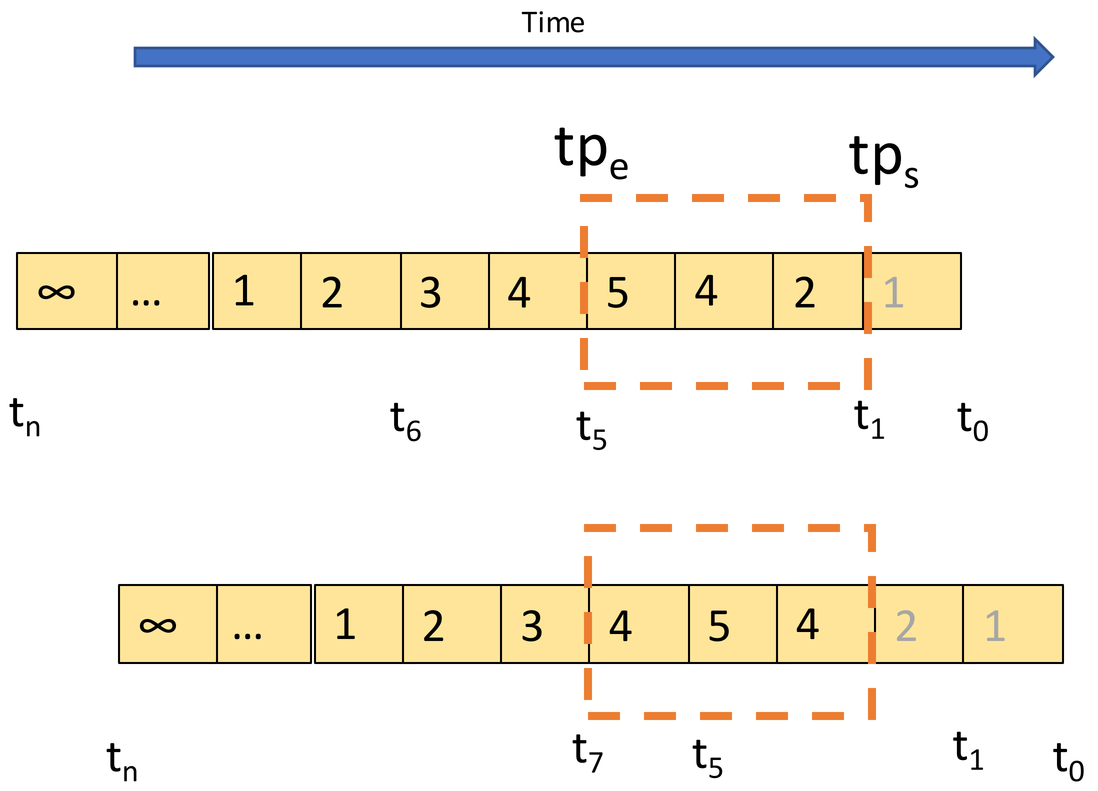
Stream Window – Time Based Window



Non-Overlapping Sliding Window

- Window size is based on time, eg 2 seconds
- Window can be advanced by:
  - t_e-t_s (window size)
  - Duration less than the window size (sliding window).
- In uniform data rate, the number of tuples will be the same for each window.

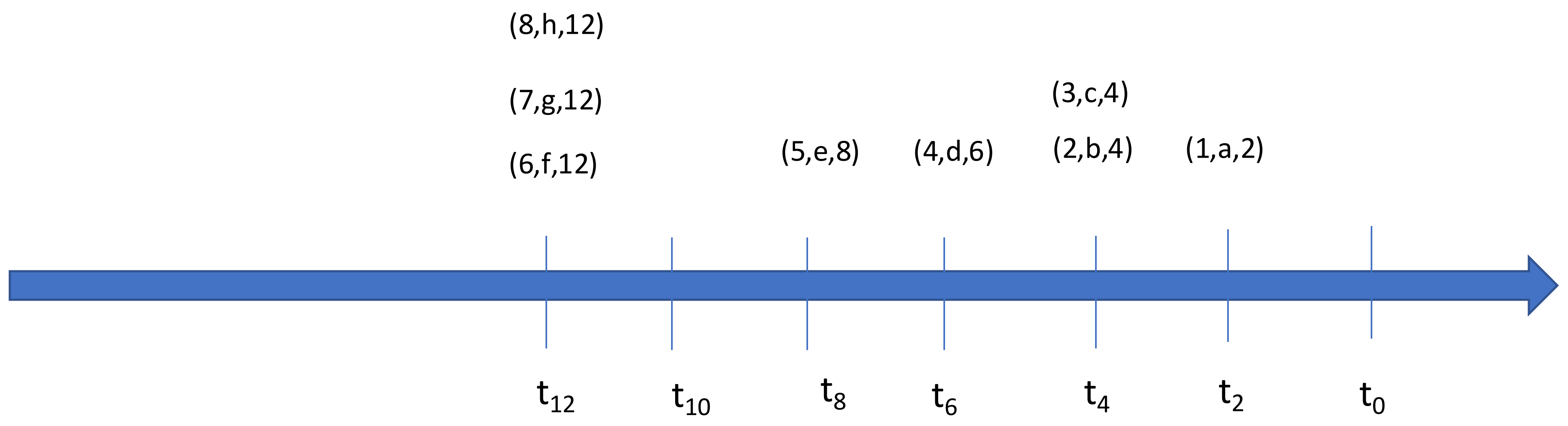
Stream Window – Tuple Based Window



Window size is 3 tuples
Slide window by 1 tuple

- ❑ Size of window is specified by number of tuples in the window
- ❑ The window always has a fixed number of tuples
- ❑ We count the number of tuples and create windows. If the window has required numbers of tuples, then that window will be processed
- ❑ The time duration of windows may vary

Example of Burst Data Arrival



processing timeline

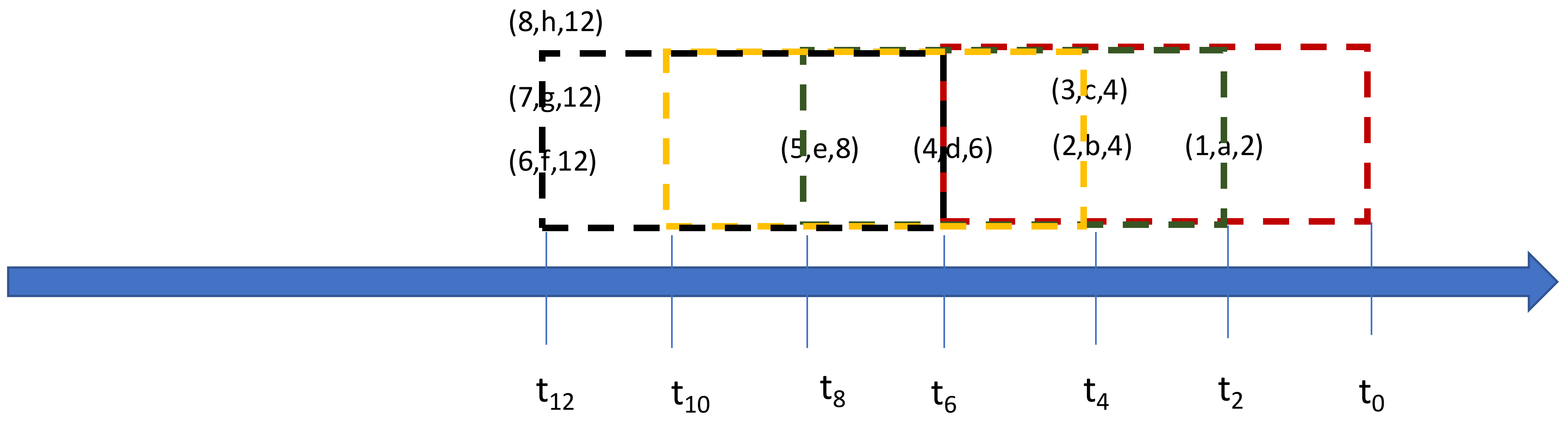
(eventTime, value, processingTime)

Event Time: The time stamp when the data is generated
Processing Time: the time stamp when the data arrived at the processing.

Time Based Window

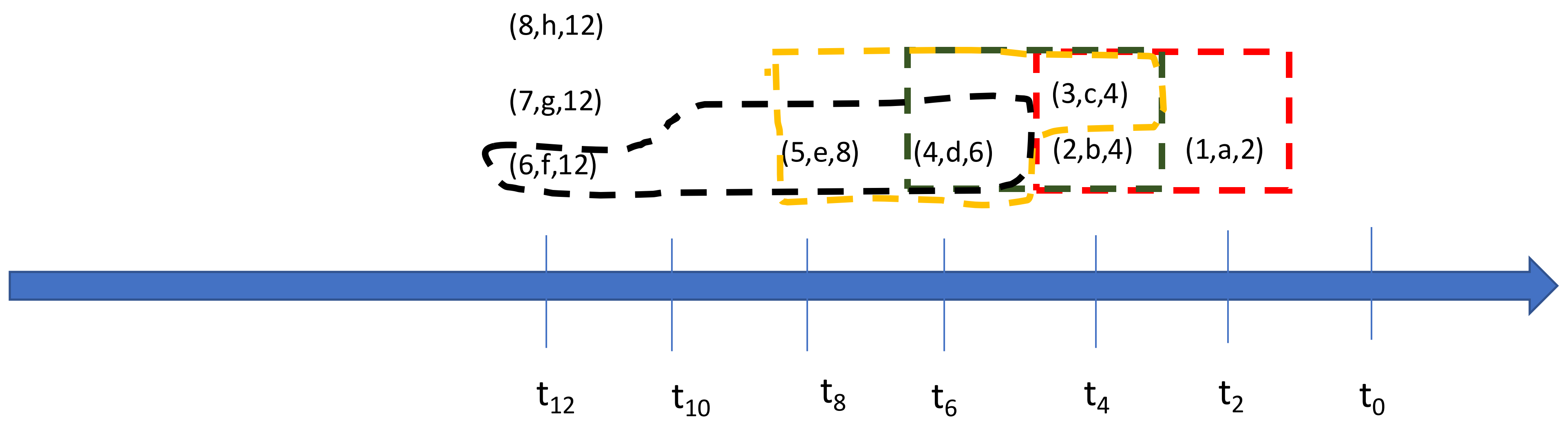
- Processing time window.
- Window size 6 time units, slide by 2 time units

W_1\{(1,a,2),(2,b,4),(3,c,4),(4,d,6)\}
W_2\{(1,a,2),(2,b,4),(3,c,4),(4,d,6),(5,e,8)\}
W_3\{(2,b,4),(3,c,4),(4,d,6),(5,e,8)\}
W_4\{(4,d,6),(5,e,8),(6,f,12),(7,g,12),(8,h,12)\}



- Duration of windows is fixed
- Number of tuples in windows can vary

Tuple Based Window



Window size is 3 tuples.
Slide by 1 tuple

W_1\{(1,a,2),(2,b,4),(3,c,4)\}
W_2\{(2,b,4),(3,c,4),(4,d,6)\}
W_3\{(3,c,4),(4,d,6),(5,e,8)\}
W_4\{(4,d,6),(5,e,8),(6,f,12)
\}

$\longrightarrow$ Need to drop some of the tuple with processing timestamp 12

- Number of tuples in windows is fixed
- Duration of windows can vary

Practice Question

A data stream application processes incoming data from a data source. The data source emits a single tuple per second (constant rate). The data are received by the data stream application in the format of *(eventTime,value)*. Once it arrives in the server, the processing timestamp is added to the tuple. The final tuple format after the addition of the processing timestamp is *(eventTimestamp, value, processingTimestamp)*.

Examples of tuples are as followed:

{1,a,1}

{2,b,4}

time-based windowing mechanism. The window size is three seconds and the slide is two seconds

{3,c,4}

{5,e,6}

{6,f,8}

{7,g,8}

{8,h,9}

{9,i,10}

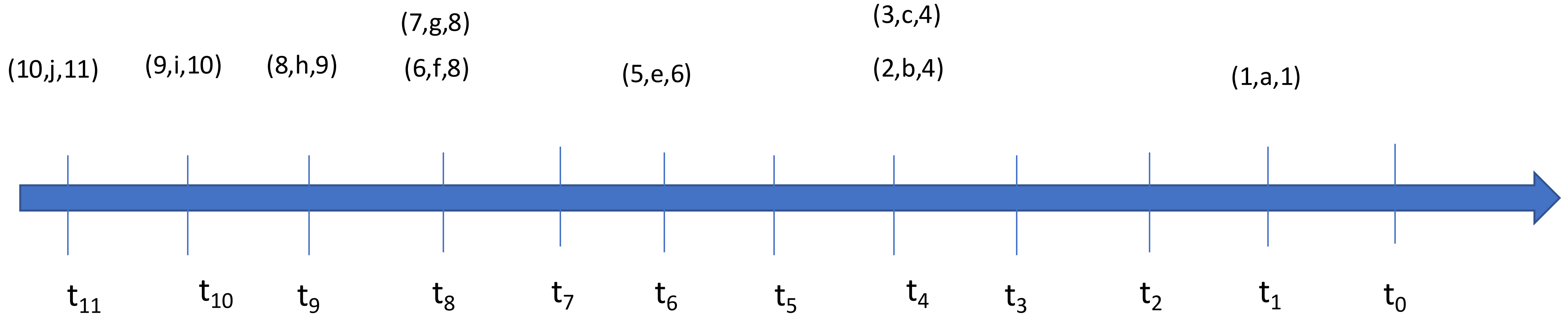
{10,j,11}

The window size is 3 seconds and the slide is 2 seconds. Assume the timing start at processing timestamp 1. What will be the content of the third window?

(eventTimestamp, value, processingTimestamp)

{1,a,1}
{2,b,4}
{3,c,4}
{5,e,6}
{6,f,8}
{7,g,8}
{8,h,9}
{9,i,10}
{10,j,11}

window size = 3 seconds
slide = 2 seconds



Processing timeline

Database vs Stream Processing

- Bounded data (assess to entire data).
- Relatively static data
- Complex, ad-hoc query
- Possible to backtrack during processing
- Exact answer to a query
- Tuples arrival rate is low

- Unbounded data (data keeps coming without end).
- Dynamic data.
- Simple, continuous query
- No backtracking, single pass operation.
- Approximate answer to a query
- Tuples arrival rate is high

Sliding Window: Produce an approximate answer to a data stream query is to evaluate the query not over the entire past history of data streams, but rather only over sliding windows of recent data from the streams.