SDCI 2012

Hot Topics in Secure and Dependable Computing for Critical Infrastructures

January 15th – 19th, Cortina d'Ampezzo, Italy

Collaborative Event Processing for the Protection of Critical Infrastructures

Time Window Based Computations: a Batch Approach

Leonardo Aniello

aniello@dis.uniroma1.it

Sapienza
Università di Roma

Dipartimento di Informatica
e Sistemistica "A. Ruberti"

MIDLAB Middleware Laboratory

Context: Cyber Attacks Detection

- Complex Event Processing (CEP)
  - enables the detection of complex attack patterns
  - huge amount of input data
  - timeliness requirements
- Time Window Based Computation (TWBC)
  - an attack takes place within a certain time window
  - we need a bound to the data to process

Time Window Based Computation (TWBC)

- a particular elaboration of a set of events that happened within a specific time window
- events are fed in the form of an event stream
- focus on
  - windows with fixed-length T
  - a new window is started ΔT after the previous window was started

APPROACHES

- **online**
  - events are processed as they enter the engine
  - many CEP engines publicly available (System S, Esper, Drools Fusion)
  - availability of mechanisms to properly manage TWBCs
- **batch**
  - events are stored as they enter the engine
  - stored events are processed periodically
  - each computation concerns a specific time window
  - MapReduce paradigm: Hadoop

HADOOP (1/2)

- distributed processing framework
- a computation is called **job** and its work is decomposed in **map** tasks and **reduce** tasks
- HDFS is the default storage, a distributed file system where data is organized in files and directories
- input data is fed to a variable set of map tasks
- they perform part of the overall elaboration and produce an halfway output
- such output is then computed by a fixed number of reduce tasks to produce the final output

HADOOP (2/2)

- limited slots for running map tasks concurrently
- if slots are exhausted, some tasks have to wait

  $\Rightarrow$increase overall computation latency
- the number of required map tasks depends on how input data is organized in files
  - in general, the higher is the number of files, the higher is the number of map tasks
  - it's better a small number of large files than a large number of small files

Why Batch instead of Online

- fault tolerance
  - if an online engine crashes, the state is lost
  - if Hadoop crashes, events are still available on HDFS
- multiple sources
  - online engines usually have a single entry point: possible bottleneck
  - in Hadoop each source can feed data to the nearest Hadoop node
- availability of online services for MapReduce computations

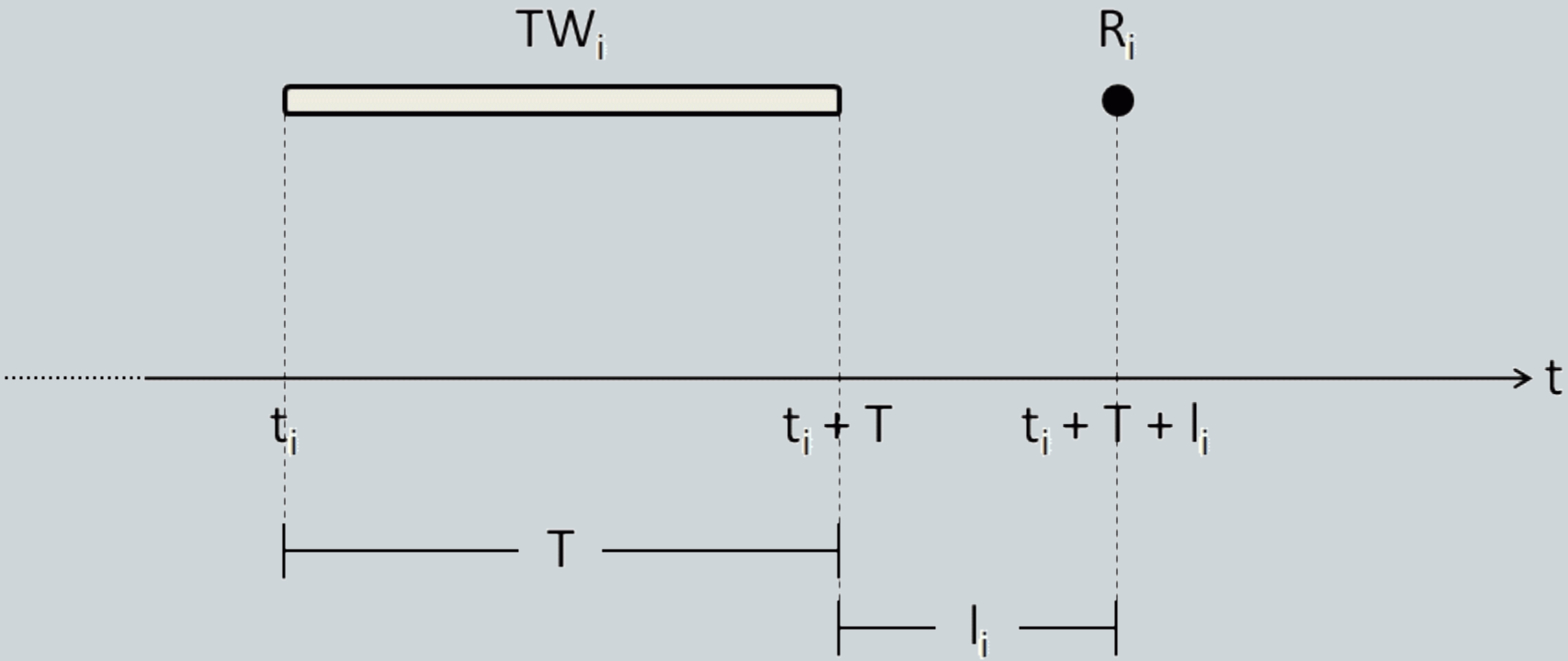
TWBCs with Hadoop

- we consider a single source
- events are written to an HDFS file as they enter the system
- once the file is closed, contained events are available for the elaboration
- **rolling strategy**

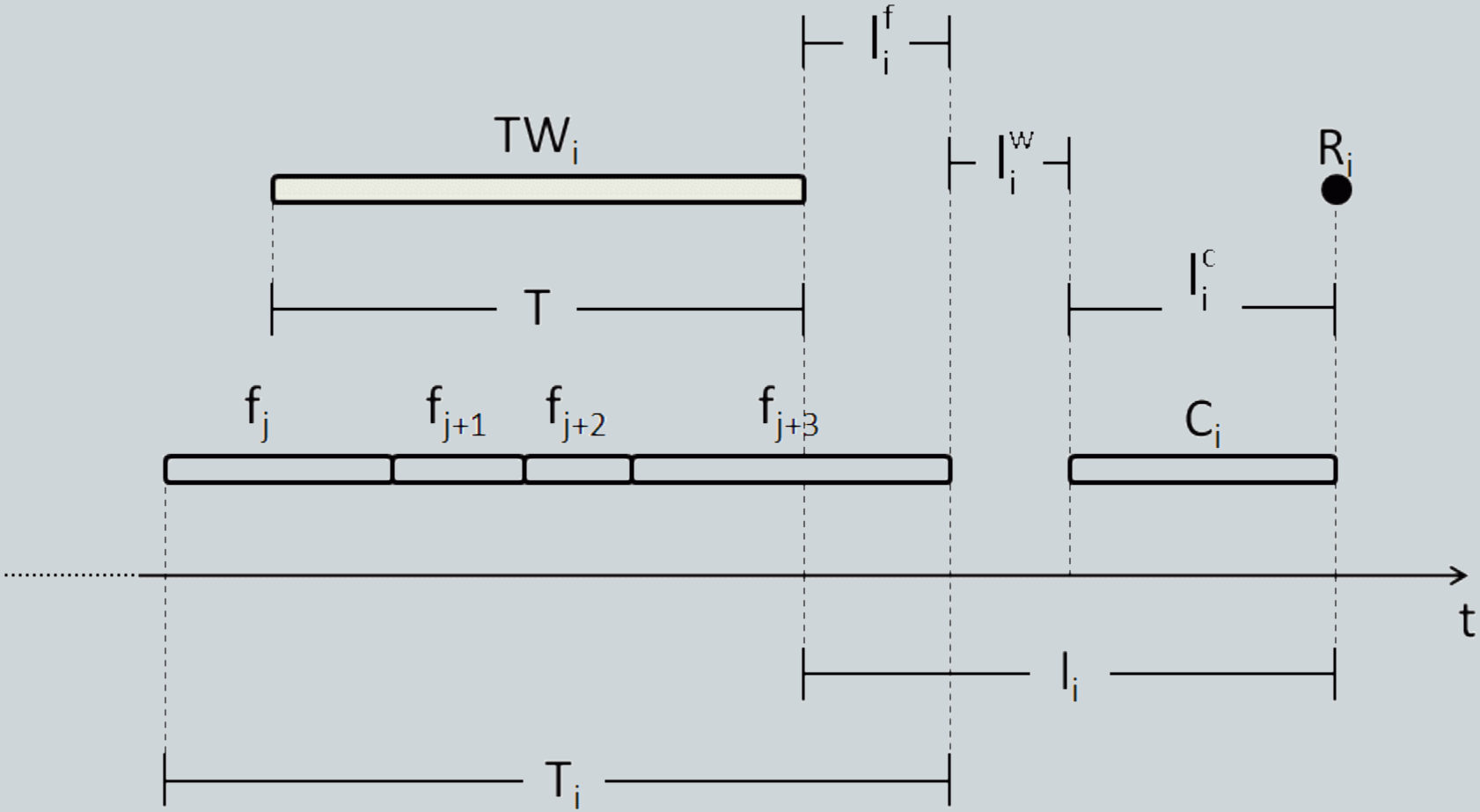
  key role of the policy used to close HDFS files, it affects
  - the number of files to process
  - the delay the events are at disposal for the job

METRICS (1/3)

- how long does it take after the end of a time window to have the result of the corresponding computation?



METRICS (2/3)



METRICS (3/3)

- **file latency**

  the time between the end of a time window and the closure of the file containing the last event of such window

- **wait latency**

  the time between the closure of the file containing the last of event of a window and the begin of the correspondent computation

- **computation latency**

- **time efficiency**

  the ratio T/T_i; it expresses the percentage of data read from file that actually belong to TW_i

Optimize the metrics (1/2)

- file latency can be minimized by closing currently open file as soon as a time window ends
- wait latency can be minimized by starting a new computation on TW_i as soon as the file containing the last event of TW_i is closed
- time efficiency can be made as near as possible to 1 by
  - opening a new file just before the first event of a time window
  - minimizing file latency

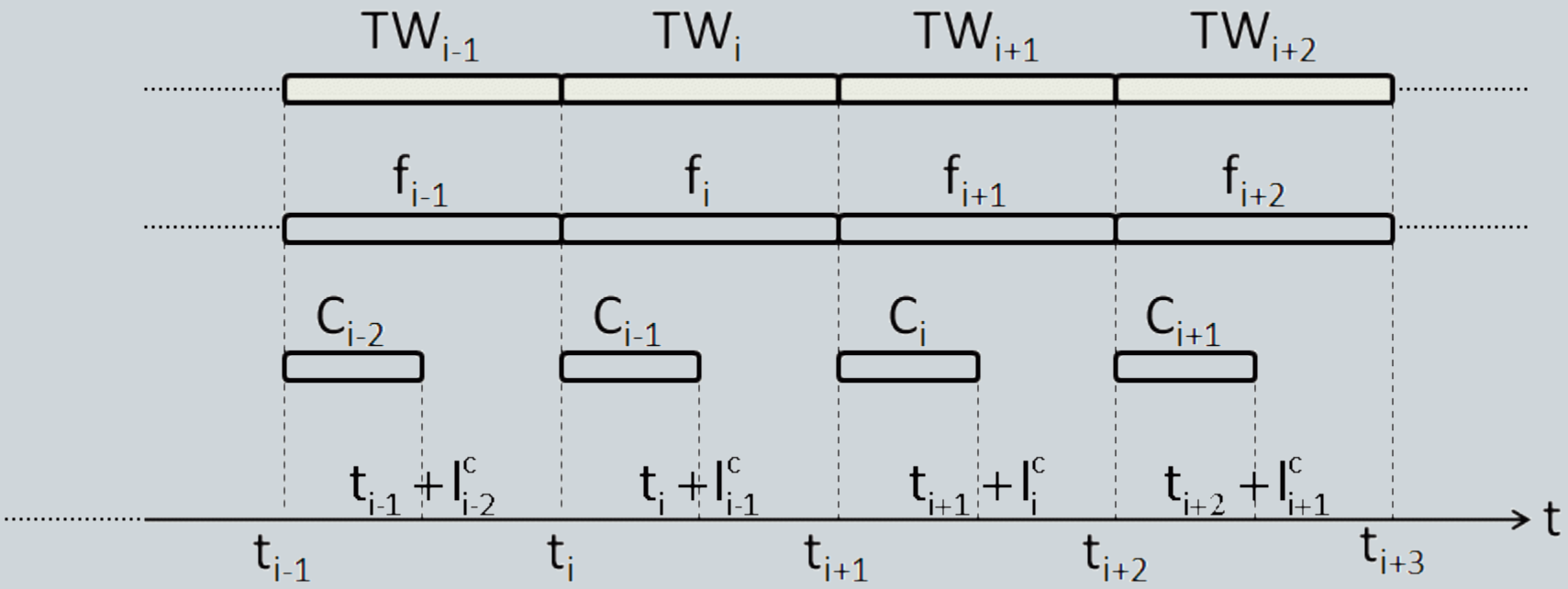
Optimize the metrics (2/2)

- how can we reduce computation latency only acting on the way we manage HDFS files?

  1. decrease the number of files used to cover a TW
     - lower overhead due to map tasks management
     - lower probabilities to run out of map slots
  2. optimize time efficiency
     - since Hadoop is IO bound, reading only what is really required helps to decrease the latency

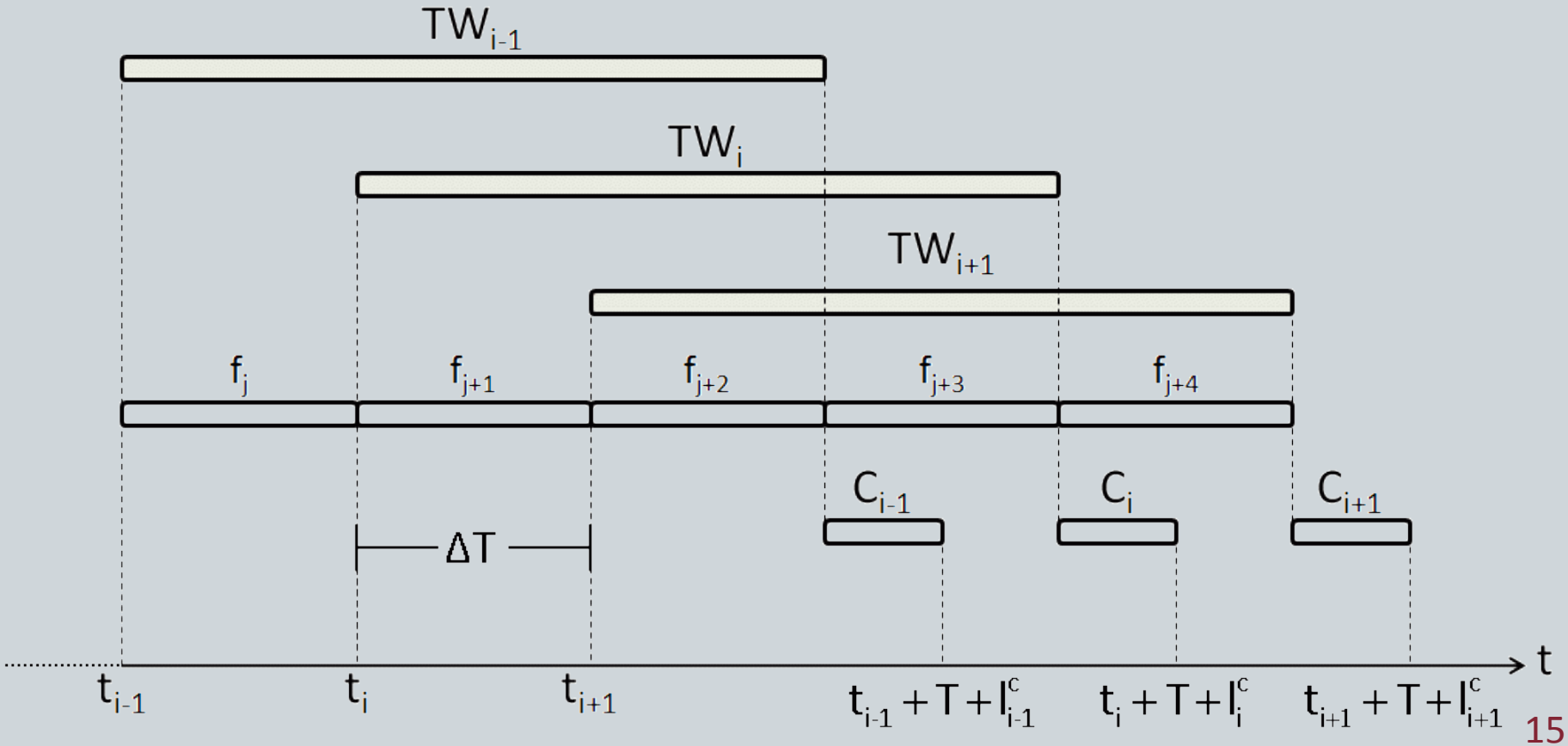
Rolling Strategy for Juxtaposed Time Windows

- $\Delta T = T$; a single file for each time window
- file latency, wait latency and time efficiency optimized
- number of files per time window optimized



Rolling Strategy for Interleaved Time Windows

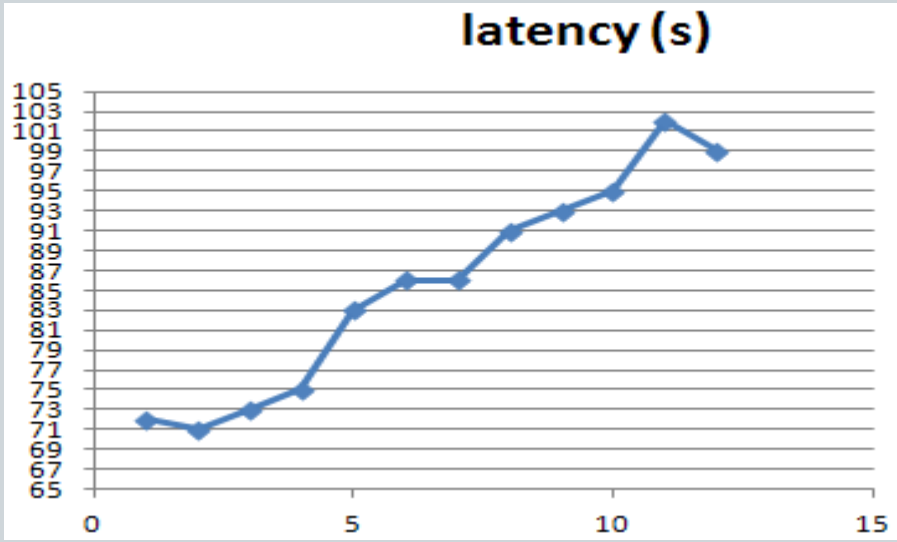
- $N \cdot \Delta T = T$; N files for each time window
- file latency, wait latency and time efficiency optimized



Rolling Strategy for Interleaved Time Windows Evaluations

- T = 1 hour; ΔT = 1 minute; N = 60
- input data: fixed packet rate
- 7 nodes Hadoop cluster + 1 data source node
- analyze the latency varying the packet rate (12 time windows)

packet rate (p/s)	average latency (s)
500	43
1000	50
2000	63

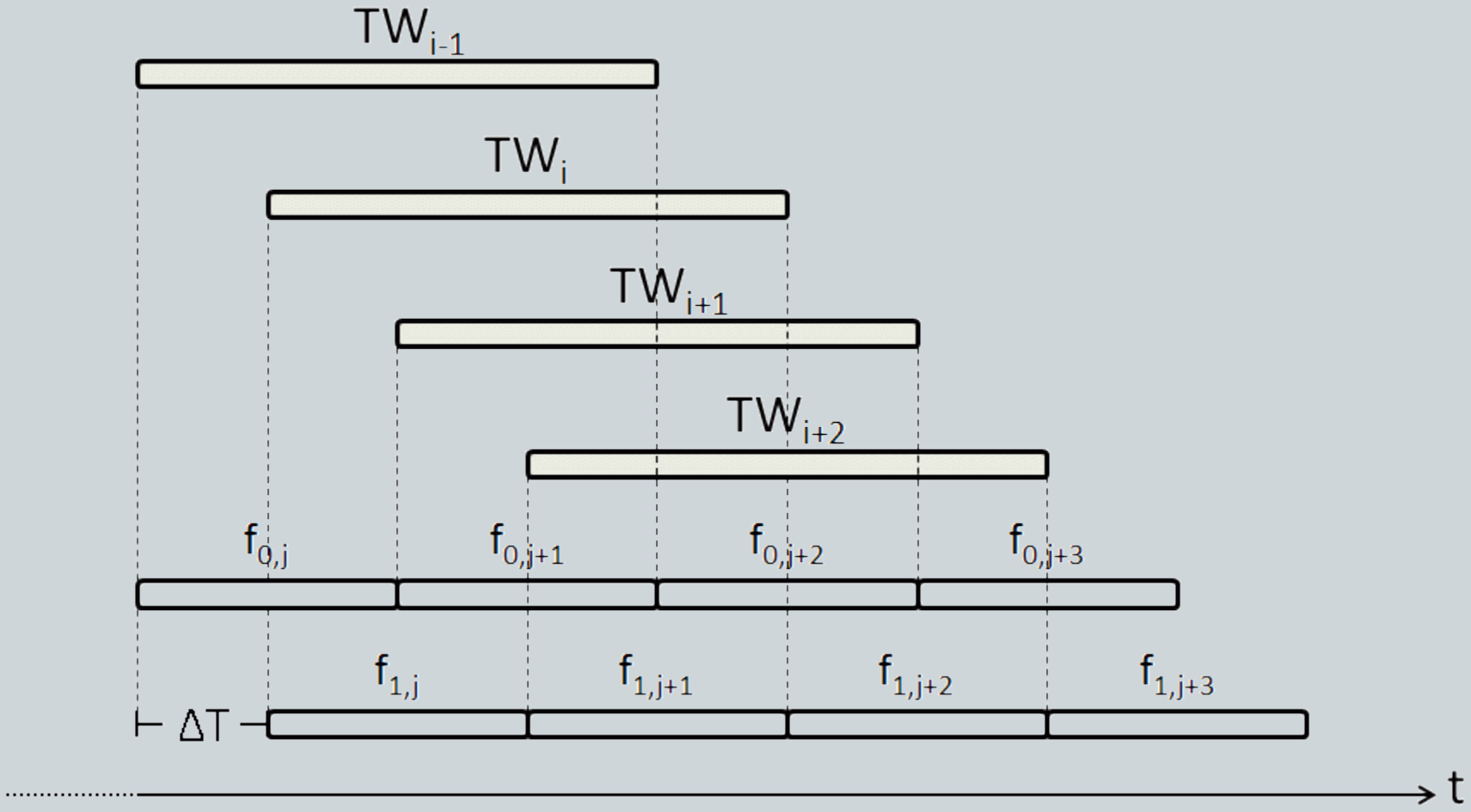


packet rate: 3000 p/s

Multi Files Flows Rolling Strategy for Interleaved Time Windows (1/4)

- to decrease the number of files per time window, we have to increase the time coverage of each file
- to keep files correctly fit time windows, so that both file latency and time efficiency are optimized, we have to use distinct files flows
- each files flow starts with an offset of ΔT from the previous one
- each time window has exactly one files flow which correctly fits

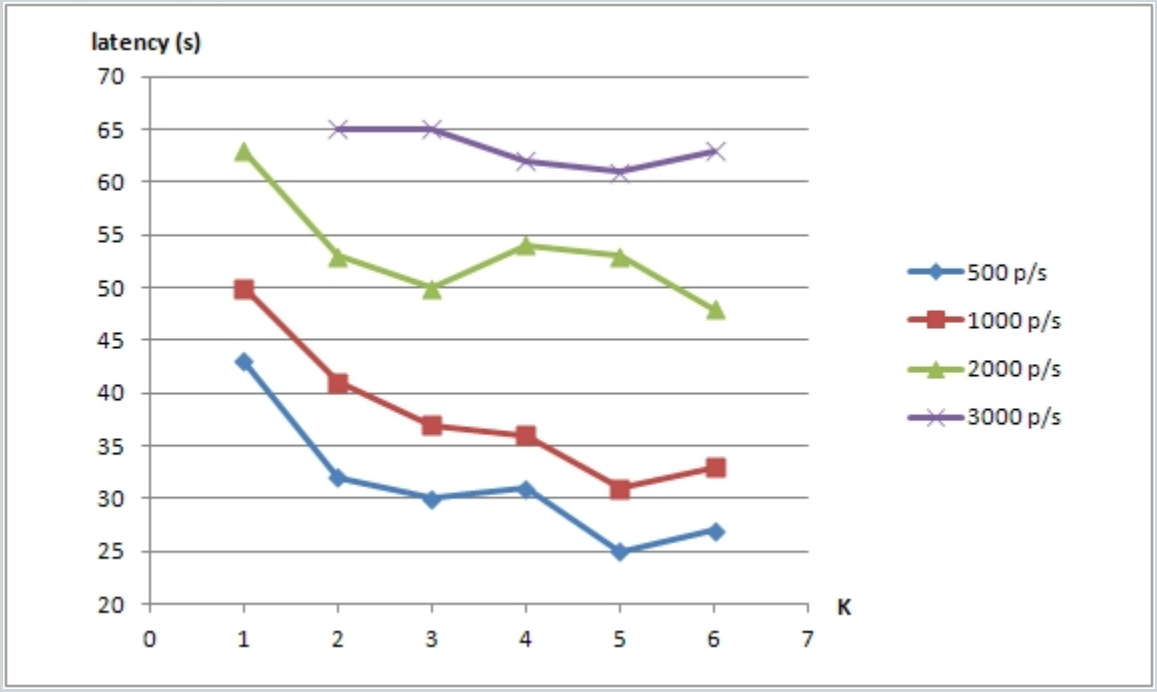
Multi Files Flows Rolling Strategy for Interleaved Time Windows (2/4)



Multi Files Flows Rolling Strategy for Interleaved Time Windows (3/4)

- K files flows (0 to K-1), where N mod K = 0
- file flow j includes files $f_{j,0}$, $f_{j,1}$, $f_{j,2}$, …
- each file covers a $K \cdot \Delta T$ time interval
- each time window is covered by N/K files
- we need more disk space (K times)

Multi Files Flows Rolling Strategy for Interleaved Time Windows (4/4)



average latency
varying input packet rate and K

Future Works

- go on with other evaluations
- consider multiple sources

Thank you!