Introduction to Run-Time Event Calculus

Manos Pitsikalis, Alexander Artikis

Institute of Informatics & Telecommunications
NCSR 'Demokritos'

manospits@iit.demokritos.gr

November 13, 2018

RTEC: Manual

The RTEC **manual** contains detailed instructions for the use of RTEC. If you have more questions refer to the **manual** ...or ask me (manospits@iit.demokritos.gr).

RTEC: Manual

The RTEC **manual** contains detailed instructions for the use of RTEC. If you have more questions refer to the **manual** ...or ask me (manospits@iit.demokritos.gr).

What is the main file of RTEC?

```
── src
    |── compiler.prolog
    |── inputModule.prolog
    |── processEvents.prolog
    |── processSDFluents.prolog
    |── processSimpleFluents.prolog
    |── RTEC.prolog
    |── utilities
        |── amalgamate-periods.prolog
        |── interval-manipulation.prolog
```

RTEC: Manual

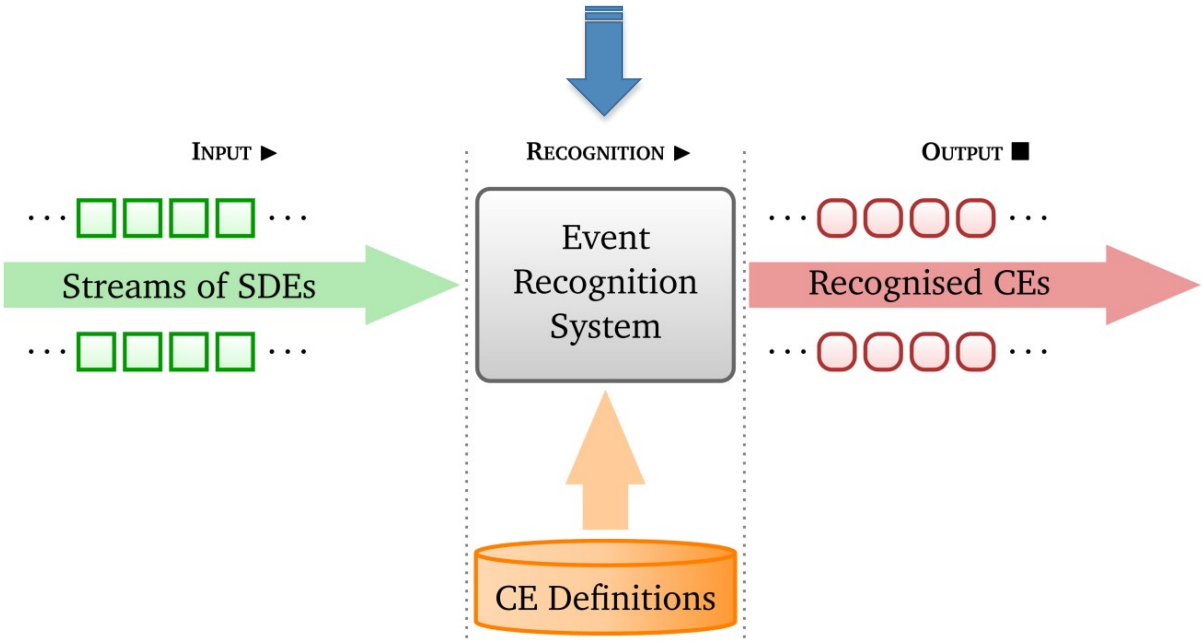
The RTEC **manual** contains detailed instructions for the use of RTEC. If you have more questions refer to the **manual** ...or ask me (manospits@iit.demokritos.gr).

What is the main file of RTEC?

```
── src
    |── compiler.prolog
    |── inputModule.prolog
    |── processEvents.prolog
    |── processSDFluents.prolog
    |── processSimpleFluents.prolog
    |── RTEC.prolog
    |── utilities
        |── amalgamate-periods.prolog
        |── interval-manipulation.prolog
```

Answer: 'RTEC.prolog'

Composite Event Recognition



Event Calculus

- A **logic programming language** for representing and reasoning about events and their effects.

Event Calculus

- A **logic programming language** for representing and reasoning about events and their effects.
- Key components:
  - **event** (typically instantaneous).
  - **fluent**: a property that may have different values at different points in time.

Event Calculus

- A **logic programming language** for representing and reasoning about events and their effects.
- Key components:
  - **event** (typically instantaneous).
  - **fluent**: a property that may have different values at different points in time.
- Built-in representation of **inertia**:
  - $F = V$ holds at a particular time-point if $F = V$ has been *initiated* by an event at some earlier time-point, and not *terminated* by another event in the meantime.

Event Recognition Engine : RTEC

- **Event Calculus for Run-Time reasoning** (*RTEC*) is the implementation of the Event Calculus designed to efficiently perform composite event recognition.
- *RTEC* recognizes from a stream of low level events the maximal intervals of durative composite events, or timepoints respectively for instantaneous events.

RTEC : Predicates

Predicate	Meaning
happensAt(E, T)	Event E occurs at time T
initiatedAt($F = V, T$)	At time T a period of time for which $F = V$ is initiated
terminatedAt($F = V, T$)	At time T a period of time for which $F = V$ is terminated
holdsFor($F = V, I$)	I is the list of the maximal intervals for which $F = V$ holds continuously
holdsAt($F = V, T$)	The value of fluent F is V at time T
union_all($[J_1, \ldots, J_n], I$)	$I = (J_1 \cup \ldots \cup J_n)$
intersect_all($[J_1, \ldots, J_n], I$)	$I = (J_1 \cap \ldots \cap J_n)$
relative_complement_all ($I', [J_1, \ldots, J_n], I$)	$I = I' \setminus (J_1 \cup \ldots \cup J_n)$

RTEC: Fluents

Fluents in RTEC can be either **simple** or **statically determined fluents**.

- **Simple fluents** are defined by specifying the initiation and termination conditions with the use of initiatedAt and terminatedAt predicates.
- **Statically determined fluents** are defined with the use of holdsFor rules.

Simple Fluents: Example

A vessel could be considered stopped when it has a speed value less than 0.5 knots.

$$\begin{aligned}
&\text{initiatedAt}(\mathit{stopped}(\mathit{Vessel}) = \text{true},\ T) \leftarrow \\
&\quad \text{happensAt}(\mathit{velocity}(\mathit{Vessel}, \mathit{Speed}, \mathit{Heading}, \mathit{CoG}), T), \\
&\quad \mathit{Speed} < 0.5. \\
&\text{terminatedAt}(\mathit{stopped}(\mathit{Vessel}) = _\mathit{PortStatus},\ T) \leftarrow \\
&\quad \text{happensAt}(\mathit{velocity}(\mathit{Vessel}, \mathit{Speed}, \mathit{Heading}, \mathit{CoG}), T), \\
&\quad \mathit{Speed} >= 0.5.
\end{aligned}$$

Statically Determined Fluents: Example

A 'naive' definition of vessels sailing with travel speed is the following:

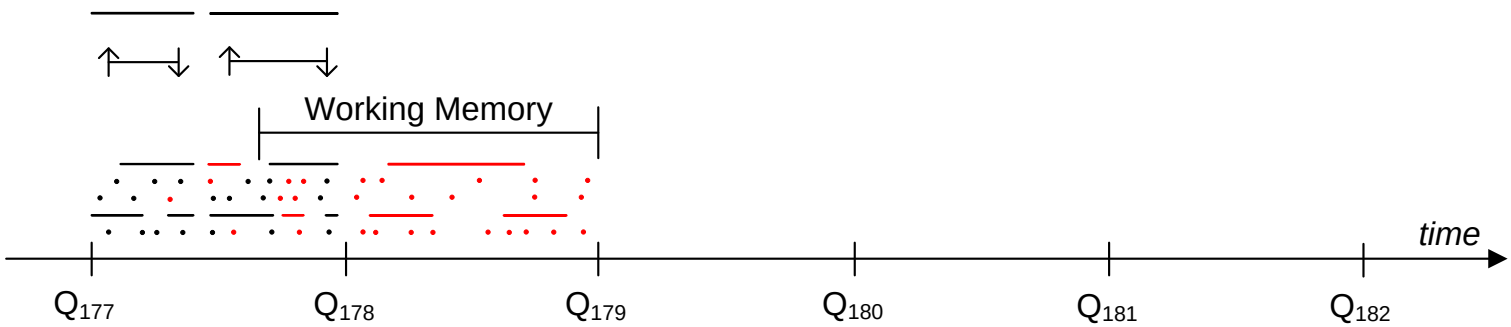
$$\begin{aligned}&\text{holdsFor}(\mathit{travelSpeed}(\mathit{Vessel}) = \text{true},\ I) \leftarrow \\ &\quad \text{holdsFor}(\mathit{underWay}(\mathit{Vessel}), I_u) \\ &\quad \text{holdsFor}(\mathit{lowSpeed}(\mathit{Vessel}), I_l) \\ &\quad \text{relative\_complement\_all}(I_u, [I_l], I).\end{aligned}$$

RTEC recognition parameters

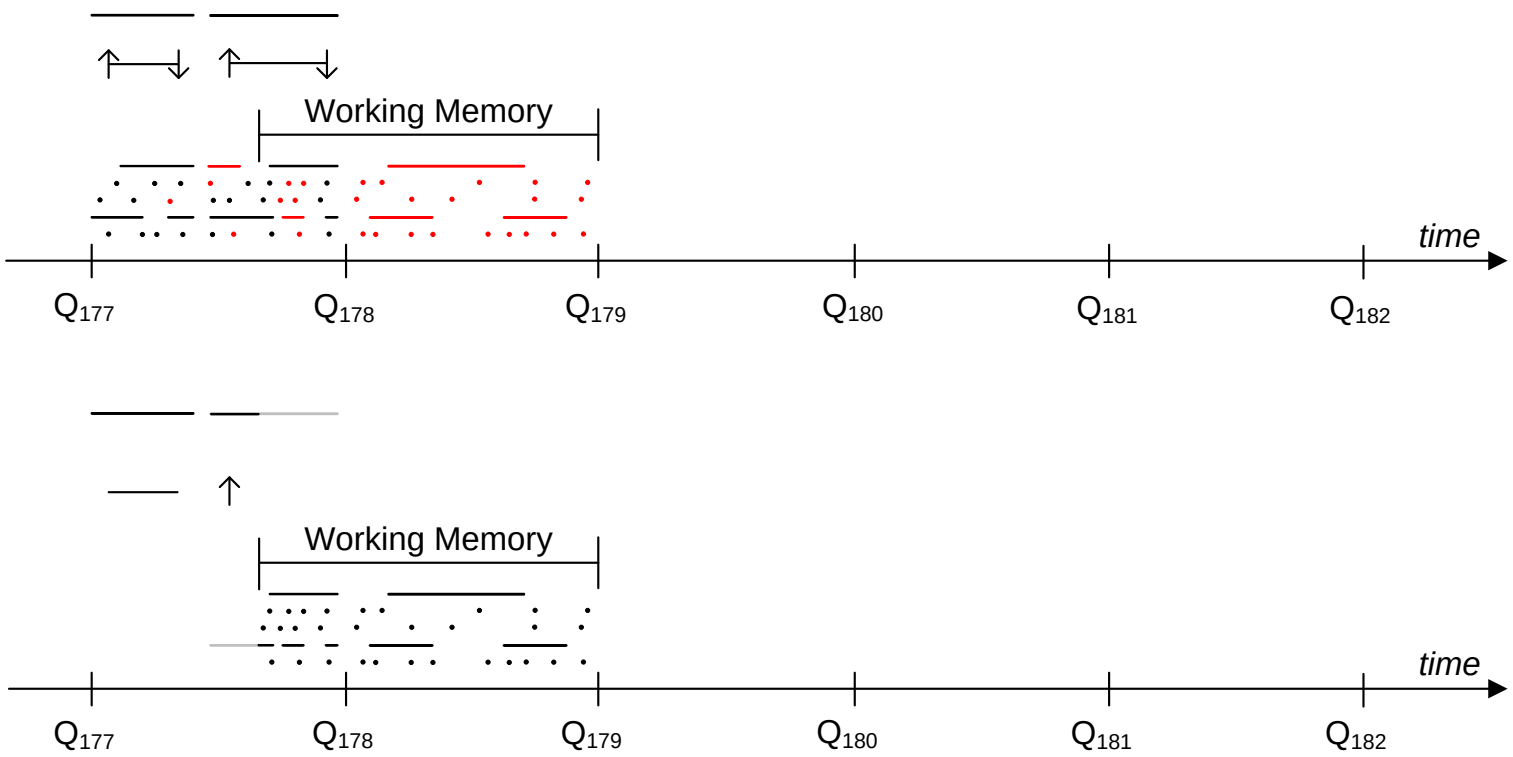
RTEC performs recognition using a window mechanism i.e., working memory.

- **Step**: CE recognition takes place at query times $Q_1, Q_2, \ldots, Q_i, \ldots, Q_n$ where $Q_i - Q_{i-1} = Step$ is a constant value.
- **WM**: At each Q_i RTEC computes the maximal intervals of CEs using input events that fall within the interval $(Q_{i-WM}, Q_i]$. Input events that occur before Q_{i-WM} are discarded.

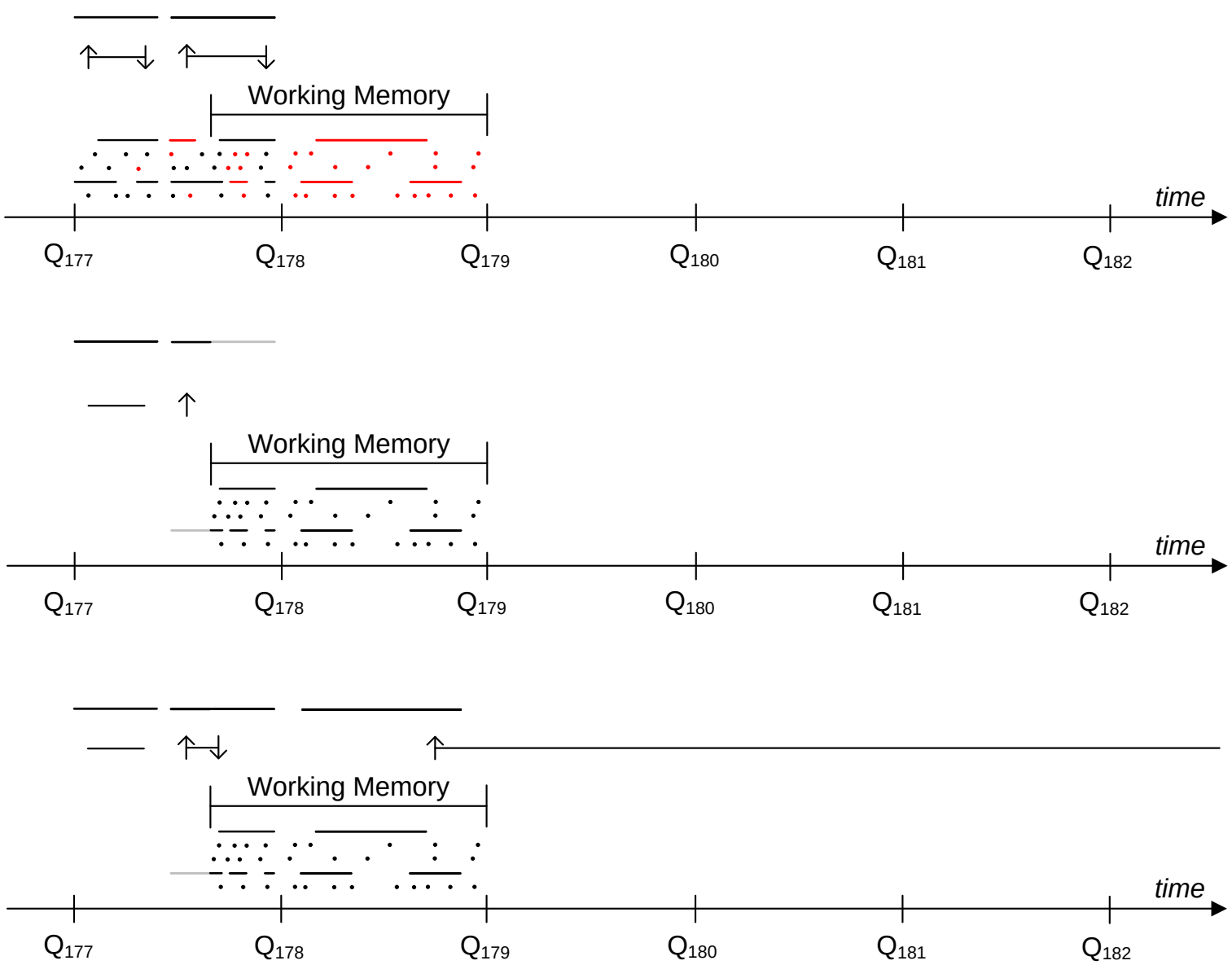
RTEC: Windows



RTEC: Windows



RTEC: Windows



RTEC: Main files

- A **patterns file** that contains rules written in the language of RTEC.
- A **declarations file** that contains the appropriate declarations for all the input and output entities.
- The **'RTEC.prolog'** is the main file of the RTEC implementation. It is required both for the compilation of rules and the recognition process.
- The **'continuousQueries.prolog'** file is used for performing event recognition.

RTEC: Patterns file

```
initiatedAt(stopped(Vessel)=true,T):-
    happensAt(velocity(Vessel,Speed,_CoG,_TrHeading),T),
    Speed < 0.5.   %knots

terminatedAt(stopped(Vessel)=true,T):-
    happensAt(velocity(Vessel,Speed,_CoG,_TrHeading),T),
    Speed >= 0.5.  %knots

initiatedAt(lowSpeed(Vessel)=true,T):-
    happensAt(velocity(Vessel,Speed,_CoG,_TrHeading),T),
    Speed >= 0.5,  %knots
    Speed < 5.0.   %knots

terminatedAt(lowSpeed(Vessel)=true,T):-
    happensAt(velocity(Vessel,Speed,_CoG,_TrHeading),T),
    Speed < 0.5.   %knots

terminatedAt(lowSpeed(Vessel)=true,T):-
    happensAt(velocity(Vessel,Speed,_CoG,_TrHeading),T),
    Speed >= 5.0.  %knots
```

RTEC: Declarations file

- For each entity state if it is input or output (simple fluents are by definition output entities), and state its index.

```
event(velocity(_,_,_,_)).
inputEntity(velocity(_,_,_,_)).
index(velocity(Vessel,_,_,_), Vessel).
  ...
simpleFluent(stopped(_) = true).
outputEntity(stopped(_) = true).
index(stopped(Vessel) = true, Vessel).
  ...
sDFluent(travelSpeed(_)=true).
outputEntity(travelSpeed(_)=true).
index(travelSpeed(Vessel)=true, Vessel).
```

RTEC: Declarations file

- For each entity state if it is input or output (simple fluents are by definition output entities), and state its index.

```
event(velocity(_,_,_,_)).
inputEntity(velocity(_,_,_,_)).
index(velocity(Vessel,_,_,_), Vessel).
  ...
simpleFluent(stopped(_) = true).
outputEntity(stopped(_) = true).
index(stopped(Vessel) = true, Vessel).
  ...
sDFluent(travelSpeed(_)=true).
outputEntity(travelSpeed(_)=true).
index(travelSpeed(Vessel)=true, Vessel).
```

- Declare the groundings of the fluents and output entities/events.

```
grounding(stopped(Vessel) = true)     :- vessel(Vessel).
  ...
grounding(travelSpeed(Vessel) = true) :- vessel(Vessel).
```

RTEC: Declarations file

- For each entity state if it is input or output (simple fluents are by definition output entities), and state its index.

```
event(velocity(_,_,_,_)).
inputEntity(velocity(_,_,_,_)).
index(velocity(Vessel,_,_,_), Vessel).
  ...
simpleFluent(stopped(_) = true).
outputEntity(stopped(_) = true).
index(stopped(Vessel) = true, Vessel).
  ...
sDFluent(travelSpeed(_)=true).
outputEntity(travelSpeed(_)=true).
index(travelSpeed(Vessel)=true, Vessel).
```

- Declare the groundings of the fluents and output entities/events.

```
grounding(stopped(Vessel) = true)     :- vessel(Vessel).
  ...
grounding(travelSpeed(Vessel) = true) :- vessel(Vessel).
```

- A caching order should be defined for all output entities. Caching order is the order in which output entities are processed.

```
cachingOrder(stopped(_) = true).
  ...
cachingOrder(travelSpeed(_) = true).
```

RTEC: Preparation

In order to perform event recognition with RTEC, first you must compile the rules...

```
% open yap prolog
['RTEC.prolog']. %load RTEC file
compileEventDescription('./declarations.prolog',      % path to declarations file
                        './patterns.prolog',          % path to patterns file
                        './compiled_patterns.prolog').% path to compiled patterns
```

RTEC: Preparation

In order to perform event recognition with RTEC, first you must compile the rules...

```
% open yap prolog
['RTEC.prolog']. %load RTEC file
compileEventDescription('./declarations.prolog',      % path to declarations file
                        './patterns.prolog',          % path to patterns file
                        './compiled_patterns.prolog').% path to compiled patterns
```

...create a dataset in the appropriate form...

```
%Event|T|T|Arg1|...|ArgN
coord|1455926402|1455926402|227091000|-4.478240000|48.383200000
%Fluent|Te|Ts|Te|Value|Arg1|...|ArgN
proximity|1455926423|1455926423|1455926423|true|227091000|227574020
```

RTEC: Preparation

In order to perform event recognition with RTEC, first you must compile the rules...

```
% open yap prolog
['RTEC.prolog']. %load RTEC file
compileEventDescription('./declarations.prolog',      % path to declarations file
                        './patterns.prolog',          % path to patterns file
                        './compiled_patterns.prolog').% path to compiled patterns
```

...create a dataset in the appropriate form...

```
%Event|T|T|Arg1|...|ArgN
coord|1455926402|1455926402|227091000|-4.478240000|48.383200000
%Fluent|Te|Ts|Te|Value|Arg1|...|ArgN
proximity|1455926423|1455926423|1455926423|true|227091000|227574020
```

...create a file with the grounding domains...

```
vessel(227091000).
vessel(227574020).
  ...
```

then...

RTEC: Recognition

Load the necessary files and perform recognition.

```
['continuousQueries.prolog']
['declarations.prolog'].
['compiled_patterns.prolog'].
['vessels.prolog'].

% continuousER(
%    DatasetFile,  DatasetFile is the input dataset file
%    OutputFile,   OutputFile records recognised CEs
%    TimesFile,    TimesFile records the event recognition times,
%    InputFile,    InputFile records the number of input events per window,
%    InitPoint,    InitPoint is where recognition starts,
%    LastTime,     LastTime is where recognition ends,
%    WM,           WM is the window size,
%    Step          Step is the recognition step.
%  )

continuousER('dataset.txt','recognised_CEs.txt',
             'stats_times.txt','stats_input.txt',
             1455926402,1456358403,86400,86400).
```