WEB SIMULATION AS A PLATFORM FOR TRAINING SOFTWARE APPLICATION

Stepan Kartak

Faculty of Electrical Engineering and Informatics, University of Pardubice

stepan.kartak@student.upce.cz

ABSTRACT
This paper is focused on a practical use of a web browser in a web distributed simulation. A web browser environment is used for a realization of a comprehensive solution for multiuser trainer applications. A computer with the web browser as one simulator station is usually available anywhere. Presented solution allows to create a script of simulation, to run the simulation and then to centrally monitor status of individual stations and behavior of their operation in real time. The individual stations can be in one room or they can be geographically distant. All this wherever and without a need of special software. An internet browser and PC connection to a PC network is the only requirement.

Keywords*:* distributed simulation, web-based simulation, WebRTC, software for training

1. INTRODUCTION
This contribution deals with the realization of the multiuser trainer applications just with an usage of web technologies. A base of the solution is based on a previous work regarding distributed web simulation (Kartak and Kavicka 2014; Hridel and Kartak 2013), that has solved only WebRTC applicability without an additional user-friendly control and software support of a remote control and central visualization. The term “trainer application” will represent a group of applications (an interactive simulation) for testing (examination, teaching) workers/dispatchers. The dispatcher (i.e. one logical process that is operated by one user at one computer/station). This user operates above specific area (then without knowledge of a global status) and responds to incoming information from nearby logical processes with which it is in direct connection. For instance let us state an operation (i.e. dispatcher) of a railway station within a region (where more dispatchers occur). Another example a management of road transport can be within an exposed territorial entity, a management of investigative, life-saving or military operations under geographical areas. Individual participants of the simulation can be in one room or they can be geographically distant (for example they need not move away from their real workplaces). PC connected to the PC network is the only requirement.

2. CHARACTERISTIC
Entire solution consists of four parts which will be described in detail in next chapters:

- administration web interface (AWI) for assembly and setting of the simulation,
- web application for central mass management of pages in browser (JSRC) – in this case it will be used for a management of logical processes,
- logical processes (LP) realized as independent web pages put above common JavaScript library,
- central visualization (CV) of individual logical processes and of entire simulation.

Due to a character but found WebRTC weak points too (see below) the entire solution focuses on distributed simulation of smaller range, in this case to 20 logical processes, however this restriction is given by current state of WebRTC only. Theoretically it is not a problem to operate a simulation of much larger range.

2.1. All in web browser
All parts are realized as web applications running in web browser (with a server support for partial additional activities – AWI, record of animation activities, etc.). Very distributed simulation then runs fully in web browsers (without using the server). A support of new HTML5 technologies (i) WebRTC (full-duplex network communication between browsers) is essential, (ii) WebSockets (full-duplex communication server-browser), (iii) canvas (medium for 2D drawing to a display area of browser) and (iv) WebGL (using of graphic card for 3D animation in the environment of web browser). These features support browsers using web core Webkit (primarily Google Chrome and many others) and Gecko (Mozzila Firefox) in both cases both desktop and mobile version. For the distributed web simulation first two above mentioned features are key which allow to communicate on network as well as standard desktop applications.
Other two features are necessary for realization of a graphic output.

2.2. Web browsers and communication on network

Due to a request for fluent animation output it is necessary to minimize network response among logical processes and it is important too to consider a fact that web browser and its programming language JavaScript do not belong among computationally efficient languages or environments. Too an option of suitable communication and synchronization of course among logical processes is essential. WebSockets communicate on basis of TCP connection and permit connection with server. This possibility would be potentially suitable for a realization of distributed simulation solved similarly as HLA (it very often selects using of centralized architecture; Kuhl, Judith, and Weatherly 2000). Here a summarizing of information on one node in the network and therefore creating of global state space of running simulation is a significant advantage of using server element. In this case we need to send a message via network twice for delivery of the message (these are four messages with a request to reply). An operation is a result that is very time-consuming and reduces possibilities of creating above mentioned type of application. WebRTC technology is a better choice since it is proposed for creating direct connection among browsers (i.e. it allows to omit "center" server element from the communication among logical processes). Network response time will be two times shorter theoretically when using WebRTC compared to WebSocket. The omitting of center element is a negative aspect as it results in a loss of simple realization of global state space (for example for a central accounting, a simulation recapitulation or for yet mentioned central visualization).

2.3. Using and problems of WebRTC

In theory connections through WebRTC are not limited in no way. It does not depend on a network topology, local networks, NAT routers (described in detail in Kartak and Kavicka 2014) not even on a number of connections. From this view it is possible to create a web alternative of a Distributed Interactive Simulation (DIS) standard in the sense of broadcast UDP communication (IEEE standards). For creating of connection in this style it is necessary to realize WebRTC peer-to-peer connection of each with each which should not be a problem in theory due to above mentioned "limitless" situation. At an extensive (connection of twenty or more logical processes) practical tests of this solution it revealed that a problem appears with frequent closing connections without an obvious reason right after connection. It is possible in the local network to create the connection of each with each among twenty computers however usually it is a need of many trials (even tens of trials with increasing number of browsers). It is a significant problem whereas after disconnection it is not possible to restore the network WebRTC connection (it depends on a type of disconnection but it is not possible in the case of above mentioned problem) without restart of application (i.e. must page refresh in the web browser). This problem identified in local site will start to show much more at "connecting" of more browsers (usually after NAT routers) across Internet public network. Even though this status is highly unsatisfactory and largely limiting for application in practice it can be to minimalize number of logical processes connections on the basis of their natural dependences among each other (see figure 1 or 3). It revealed from testing in local site that connection of 20 browsers (logical processes) over ca. 40 WebRTC connections provides stable achievable solution in the local site. As regards connection through public site problems with self-closing connections still occur however achieving all stable connections seems as possible contrary to above mentioned alternative in broadcast connection.

For an example, an attempt to connection among 6 browsers (each with one logical process, in different geographically distant net, 1 km to 150 km beeline amongst physical location of browsers) failed to repeat even after ten trials because connection between 1 or 2 browsers (usually those that were geographically furthest apart) always closed itself after connection without an apparent reason. Connection with the same topology within an urban network (geographic distance between physical location of browsers around 1 km) succeeded repeatedly among 1 and 3 trials. Even this result is not optimal. After successful connection establishment the connection is always permanent and smooth (max. 5 operation hours tested).

2.4. Synchronization of logical processes

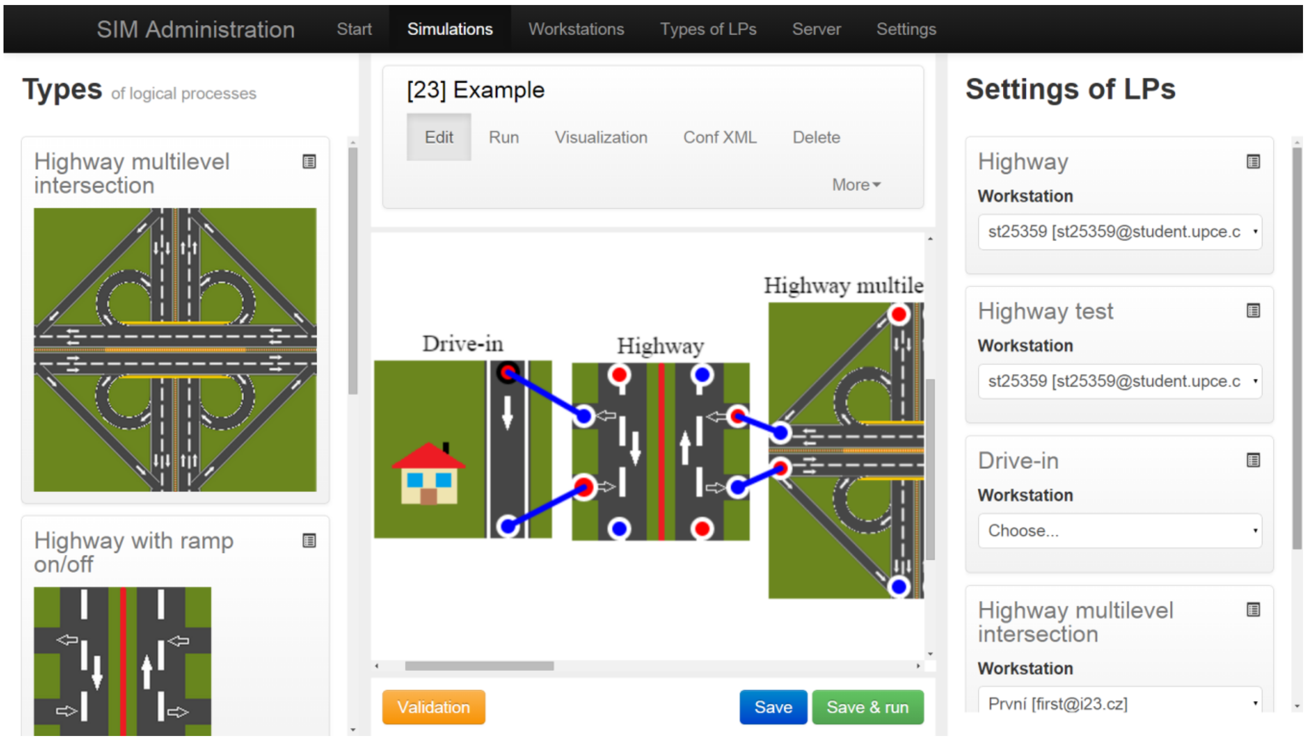
The synchronization of logical processes is realized by a method Conservative synchronization technique of sending null messages with a lookahead (Chandy-Misra-Bryant Distributed Discrete-Event Simulation Algorithm, Fujimoto 2000). It is about standard slightly modified version of this algorithm, the version was described in detail in Kartak and Kavicka 2014.

2.5. JavaScript and computational demands

Programming language JavaScript is almost exclusively one-thread in the web browser (it generally valids for entire user source code), this means that all operations are performed gradually, by time (or order) that they have attributed by internal scheduler or in nearest possible time if at the time on which an event is planned already some calculation runs. This is a critical issue with that it is necessary to count.

The logical process performs three operation groups:

- calculations of simulation core,
- realization of the animation,
- and a whole range of minor tasks (for example sending information to the server).



Highway
Workstation: st25359 [st25359@student.upce.c

Highway test
Workstation: st25359 [st25359@student.upce.c

Drive-in
Workstation: Choose...

Highway multilevel intersection
Workstation: První [first@i23.cz]

Figure 1: Administration web interface, visual editor

Just the realization of the animation currently appears as a problematic area.
Demand depends on area for drawing size (HTML element *canvas*) and too on the quantity of animation activities that it is necessary to update in each step of the animation – whereas it is about drawing here (in our case in 2D space) we will not avoid often counting of parametric curves and else not fully trivial calculations that will begin to occupy precious time in larger quantities with next operations of animation activities (time control of beginning and end of display, etc.). Generally speaking computationally intensive tasks are not an ideal type of problem for JavaScript solution and they may represent significant problem for fluency of simulation run from the perspective of service and in a combination with one-thread access (in fact it can be only about decrease of framerate of animation output). If we take into account that a man considers fluent animation it is possible at 25 FPS (generally accepted number, films are distributed with that FPS) we will get a time to a realization of calculations between redrawing of animation scene 1000 / 25 = 40 ms. During 40 ms it must be done at worst (in brackets average measured times in ms stated, if they are average larger than 1 ms; computers in one local network, 20 logical processes, topology of connection see figure 9a):

1. Calculations of simulation core, consist of:
   (a) calculation of single activities and planning of further activities (1 ms),
   (b) waiting for a response of remote logical processes because of synchronization (in this solution it is about a request for a null message if we have n nearby logical processes then a time required to realization will be $n \times 10$ ms at worst).
2. Processing planned JavaScript tasks:
   (a) execute commands of central JSRC control,
   (b) responses to requests for null message from nearby logical processes,
   (c) internal planned operations, i.e. sending information to server, sending screenshots of animation scene, etc. (2 ms).
3. Calculations of animation module (10 ms):
   (a) sorting of animation activities,
   (b) time control of beginning and end of animation activities display.

Critical time that it is not possible to optimize in no way is 1b point (waiting for response of remote logical processes because of synchronization). On average response to this type of request lasts 10 ms when whole logical process inactively waits. The more logical processes are with that it is necessary to keep logical process as synchronized the more (in the worst case) a time can be longer between animation frames. According to presented values 3 and more nearby logical processes can mean a problem.

Standard desktop application would solve a considerable part of these operations in parallel, in JavaScript something like this is not possible so must be to pay special attention to optimizing of calculations and data structures.

3. ADMINISTRATION WEB INTERFACE

Administration web interface (AWI, figure 1) is a web application that serves primarily to a compilation of distributed simulation model and to a creation of a session of this simulation. The process of creating the distributed simulation from beginning to launch is shown in a flowchart in a figure 2.

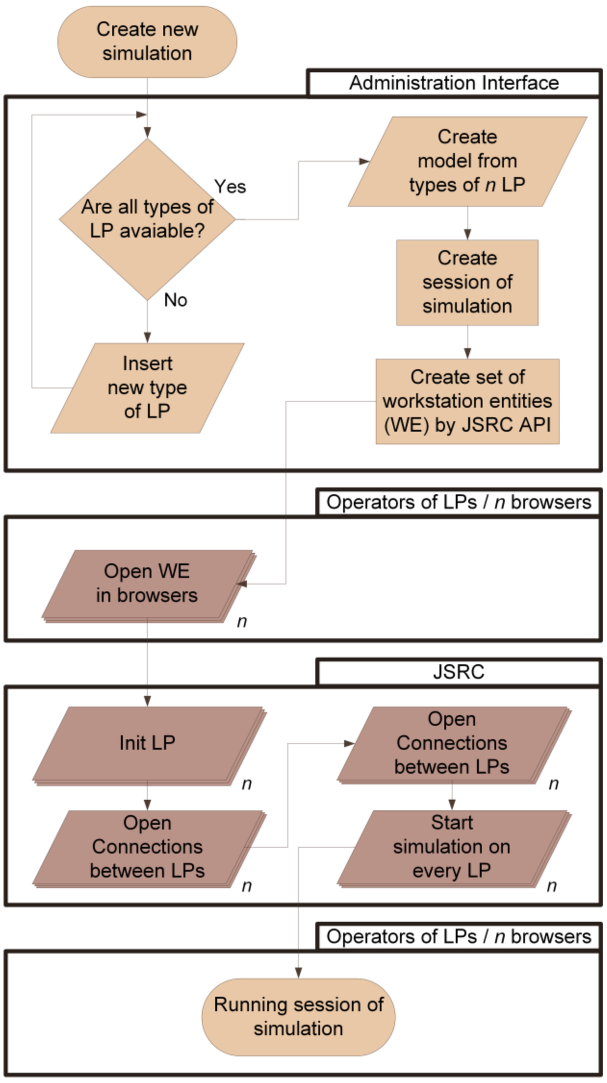

Create new simulation
Administration Interface
Are all types of LP avaiable?
Yes
No
Create model from types of *n* LP
Create session of simulation
Insert new type of LP
Create set of workstation entities (WE) by JSRC API
Operators of LPs / *n* browsers
Open WE in browsers
n
JSRC
Init LP
n
Open Connections between LPs
n
Open Connections between LPs
n
Start simulation on every LP
n
Operators of LPs / *n* browsers
Running session of simulation

Figure 2: Flowchart of process of creating the distributed simulation from beginning to launch

3.1. Logical process, its type and definition

The distributed simulation consists of logical processes. This solution considers set web page registered in AWI as the logical process. In AWI a record about new type of logical process is created. To this record (a) name, (b) URL address of very web page (it represents very logical process) and (c) URL address with XML file that specifies possibilities of setting and network linking with other logical processes are assigned. Based on this record and XML configuration in visual editor (see next chapter) a miniature of logical process is created (figure 3) that represents logical process in the editor.

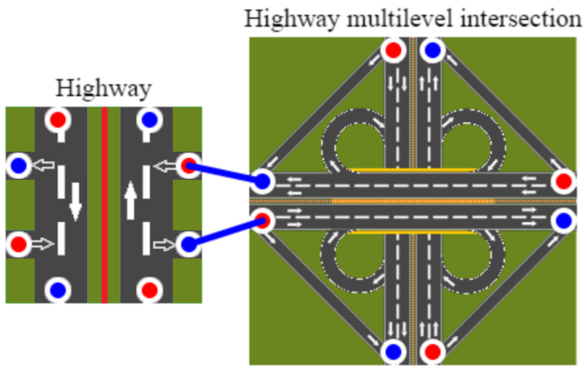


Figure 3: Display of two linked logical processes in AWI visual editor

Configuration XML file contains:

1. **Meta information –** basic information about the logical process (name, author, picture for display in AWI, etc.)

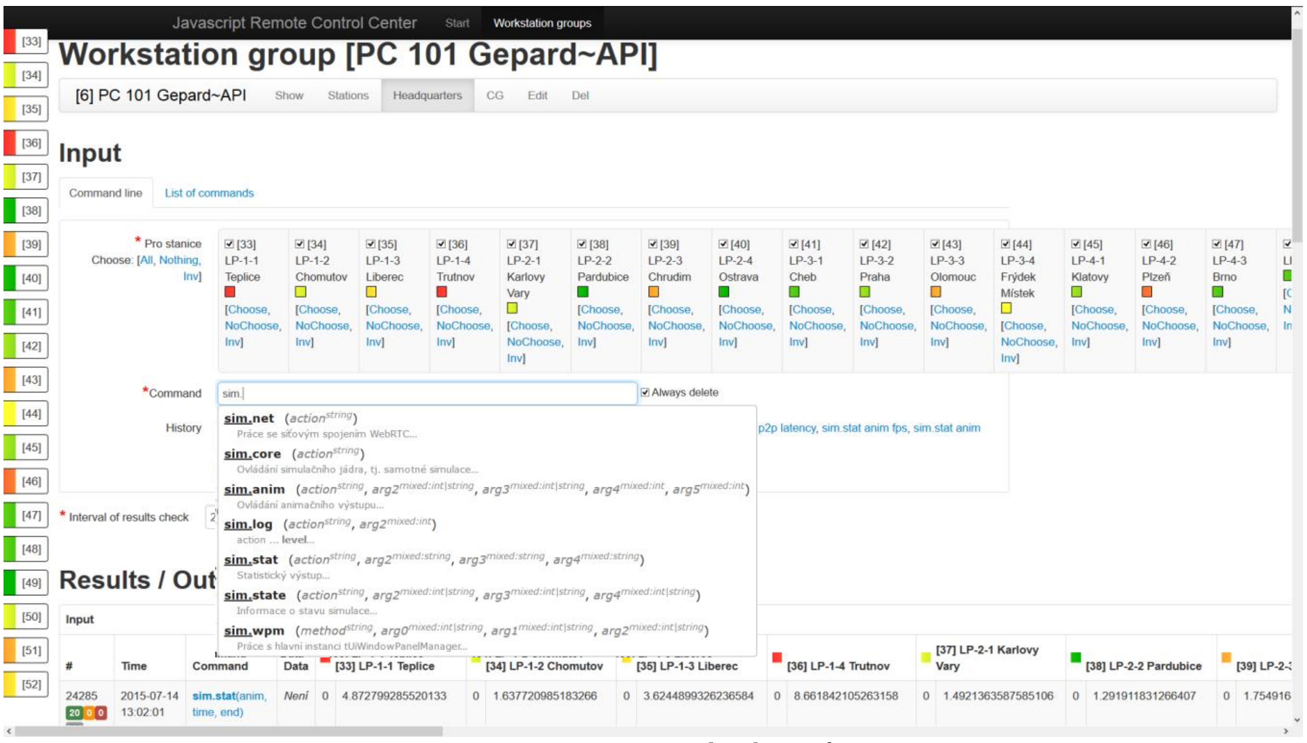
2. **Description of connection with other logical processes –** i.e. input and output entit points and their setting:
   (a) list of entit types that are valid for the logical process,
   (b) distinction of input and output points,
   (c) determining if the connection is required (usually for a meaningful function of logical process some input is required) or not.

3. **Items of setting –** each requires to specify:
   (a) name,
   (b) description,
   (c) default value,
   (d) type of value (for an appropriate processing/validation of input and display of appropriate input fields - eg *string*, *integer*, *boolean*, etc.),
   (e) and few further options (for example a key under that the logical process will expect these data).

4. **Free space** (node chosen within XML file) for any content, determined for a possibility to set logical process above editor frame options.

3.2. Visual editor for setup of distributed simulation model

The simulation model consists of a set of logical interlinked processes. Visual editor in AWI serves for creating of this set (see fig. 1).
In the left part of editor an offer of available types of logical processes is.
In the middle a design area is located that serves for setup and connecting of logical processes. The logical processes are added by dragging from a left bar. Setting and connecting of logical processes proceeds under rules said in XML configuration files of logical



Workstation group [PC 101 Gepard~API]

Input

* Pro stanice
Choose: [All, Nothing, Inv]

✓ [33] LP-1-1 Teplice 🟥 [Choose, NoChoose, Inv]	✓ [34] LP-1-2 Chomutov 🟨 [Choose, NoChoose, Inv]	✓ [35] LP-1-3 Liberec 🟨 [Choose, NoChoose, Inv]	✓ [36] LP-1-4 Trutnov 🟥 [Choose, NoChoose, Inv]	✓ [37] LP-2-1 Karlovy Vary 🟨 [Choose, NoChoose, Inv]	✓ [38] LP-2-2 Pardubice 🟩 [Choose, NoChoose, Inv]	✓ [39] LP-2-3 Chrudim 🟧 [Choose, NoChoose, Inv]	✓ [40] LP-2-4 Ostrava 🟩 [Choose, NoChoose, Inv]	✓ [41] LP-3-1 Cheb 🟩 [Choose, NoChoose, Inv]	✓ [42] LP-3-2 Praha 🟩 [Choose, NoChoose, Inv]	✓ [43] LP-3-3 Olomouc 🟧 [Choose, NoChoose, Inv]	✓ [44] LP-3-4 Frýdek Místek 🟨 [Choose, NoChoose, Inv]	✓ [45] LP-4-1 Klatovy 🟩 [Choose, NoChoose, Inv]	✓ [46] LP-4-2 Plzeň 🟧 [Choose, NoChoose, Inv]	✓ [47] LP-4-3 Brno 🟩 [Choose, NoChoose, Inv]

*Command `sim.`

History p2p latency, sim.stat anim fps, sim.stat anim

```
sim.net (action^string)
  Práce se síťovým spojením WebRTC...
sim.core (action^string)
  Ovládání simulačního jádra, tj. samotné simulace...
sim.anim (action^string, arg2^mixed:int|string, arg3^mixed:int|string, arg4^mixed:int, arg5^mixed:int)
  Ovládání animačního výstupu...
sim.log (action^string, arg2^mixed:int)
  action ... level...
sim.stat (action^string, arg2^mixed:string, arg3^mixed:string, arg4^mixed:string)
  Statistický výstup...
sim.state (action^string, arg2^mixed:int|string, arg3^mixed:int|string, arg4^mixed:int|string)
  Informace o stavu simulace...
sim.wpm (method^string, arg0^mixed:int|string, arg1^mixed:int|string, arg2^mixed:int|string)
  Práce s hlavní instanci tUiWindowPanelManager...
```

* Interval of results check 2

Results / Out

#	Time	Command	Data		🟥 [33] LP-1-1 Teplice		🟨 [34] LP-1-2 Chomutov		🟨 [35] LP-1-3 Liberec		🟥 [36] LP-1-4 Trutnov		🟨 [37] LP-2-1 Karlovy Vary		🟩 [38] LP-2-2 Pardubice		🟧 [39] LP-2-
24285 20 0 0	2015-07-14 13:02:01	sim.stat(anim, time, end)	*Není*	0	4.872799285520133	0	1.637720985183266	0	3.6244899326236584	0	8.661842105263158	0	1.4921363587585106	0	1.291911831266407	0	1.754916

Figure 4: Remote Control web interface

processes types. In the right section a quick review of used logical processes is with a possibility of their setup.

3.3. Creating of simulation session

The simulation model created as described in part 3.2 is only an executable description of a simulation structure and setup. Usable simulation became from the model by creating of simulation session (de facto of session of the simulation model).
Such action consists of these steps:

1. Assignment of an unique identifier to each logical process in the model.
2. Creating/preparing of relevant data structures:
   (a) preparing of data in a database,
   (b) creating of XML configuration file describing compilation of model with added information about simulation session (especially identifiers of single logical processes).

3. Creating of a group of stations in JSRC application (see below). Creating of single stations in this group for each logical process. (All through relevant JSRC API.)

Created XML configuration file represents enter data for very logical processes, on the basis of information from this file WebRTC connection and assistant data structures is set.

4. REMOTE CONTROL

Remote control (JSRC, web interface is shown on figure 4) is a separate application intended to a multiple control of web pages. In this case it serves to a control of logical processes. Primarily to a load of logical process, creating of WebRTC connection and startup of all logical processes.

4.1. Description of working and integration to the simulation

This application as an entirely independent product serving for a mass control of web pages that is from one place (JSRC web application) any quantity of web pages is controlled. Web pages are identified as *workstations* within JSRC.
Workstations are defined by:

- name,
- initialization page (page that will be controlled).

These workstations are grouped into categories of workstations (*workstation groups*). And just within these groups it is possible to enter mass commands (nevertheless it is possible to specify a workstation subset that will receive the command).
Really the workstation corresponds to the logical process and one group of workstations corresponds to the whole model.
The commands are entered by text form (figure 4), moreover there is an option to create advance prepared command sets at "one click" (figure 5).

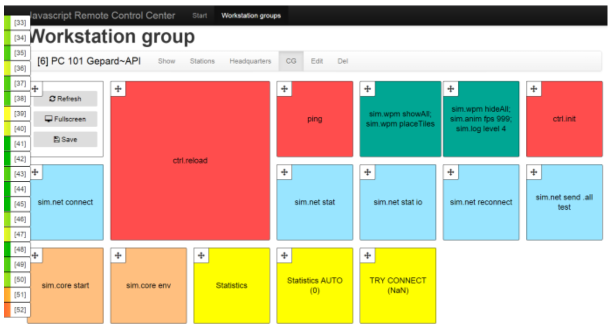


Figure 5: JSRC: Prepared command set, one square is user-defined command (or commands), prepared for touch-devices

Each work group can dispose different command set, here used for a separation of remote control of central visualization (CV, see next chapter) from very simulation.
The session of distributed simulation is realized within one work group and CV control in the second entirely independent work group (it contains only one workstation).

4.2. Description of used API

This application communicates with pages via API that allows to:

- add groups of stations (WG_ADD),
- add stations to groups (WE_ADD),
- add commands (COMMAND_ADD),
- remove commands for processing (COMMAND_FETCH),
- send responses (results) of commands (COMMAND_RESULT),
- get information about commands (primary results, COMMAND_GET),
- get information about station groups (WG_INFO),
- get information about very stations (WE_INFO).

The communication via API is displayed at picture 6.

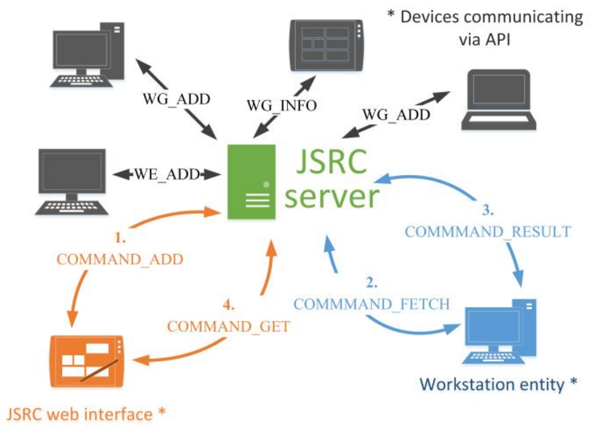


Figure 6: The communication via JSRC API with order of API calls for execute one command

An usage scheme in the simulation is displayed at figure 7.

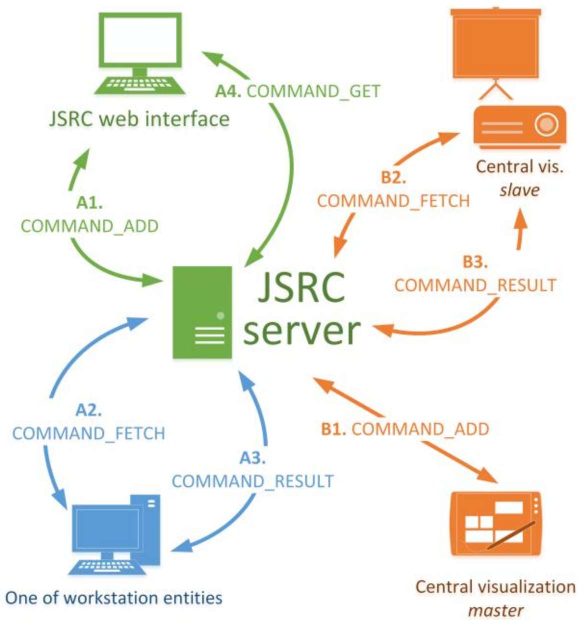


Figure 7: The scheme of usage JSRC API in the simulation

5. REALIZATION OF LOGICAL PROCESSES

The logical processes are realized as common web page in that with assistance of prepared JavaScript library (it is not a condition while respecting communication standards) is: (a) XML file for transmission of information about configuration of distributed model, (b) structure of messages transmitted among logical processes.
It can be to quickly create with this library activities for simulation cores and on the basis of several prepared prototypes of animation activities too an animation output.
While all can be created user friendly in an uniform style of control and appearance too.
So prepared web page represents type of logical process that is potentially configurable and reusable (in more models and in one model multiply too).
It is necessary to prepare a picture yet for adding new type of logical process (a miniature schematically representing the logical process) and XML configuration file specifying setup and network connections with that type of logical process disposes (described in detail in chapter 3.1).

6. CENTRAL VISUALIZATION

The central visualization (CV) is an independent web page that is created from the same components as very logical processes – especially for keeping united visual style and control. It allows to repeatedly play animation status of logical processes.

6.1. Realization of animation play of one logical process

Animation of logical processes sends new, changed and removed animation activities to the server (currently solved as a part of AWI). The sending is solved, according to server workload, in two modes: (a) sending immediately, (b) queuing of activities and their batch sending after some time. CV page in regular intervals downloads these information about animation activities that it immediately ranks to the animation output or removes them by record type. The output is due to delay in communication between logical process and server and between page with the visualization and the server burdened by delay against real in order of tens to hundreds of milliseconds (according to selected setting). Display of the communication is in a figure 7 (“b” commands).

6.2. Page with a summary of all logical processes

CV provides a possibility too to display at once status (i.e. the animation in this case) of all logical processes (within the session of distributed simulation) in bulk in the summary. Whereas fluent play of animation activities in one animation scene is computationally very difficult the realization of displaying larger number of animation scenes is difficult to achieve.

From that reason it was proceeded to that the logical processes after selected intervals (specified in XML configuration of logical process type) send to server (part of AWI) screenshots of the animation scene and these static images are then displayed in the summary – the animation is not fluent but it is enough for the summary.

Neither this solution is not ideal because creating of animation scene screenshot lasts in order of tens of milliseconds after that it is not possible to restore the animation of the logical process – so impression of choppy video generates that definitely is not an ideal status and it disturbs an user serving the logical process. Generally speaking the larger animation scene the more time creating of the screenshot will take.

6.3. Remote control of central visualization

For a comfortable CV control it is created through JSRC to control the page (primarily the animation output and selection of logical process or page with the summary) remotely from another instance/CV page.

In this case one CV is in a *master* position (i.e. control, CV_m) and the second one in a *slave* position (i.e. controlled visualization, CV_s).

A practical use CV_s display can be on a projector and its control through CV_m that can run for example on a tablet that service or trainer handles.

The animation output is always synchronized with a system time and therefore we can suppose that one second of CV_s and CV_manimation will in real time correspond to one second.

This premise is not valid relative to JavaScript “one-thread” architecture and displayed time at both animations will not remain synchronized for a long time.

From this reason it was proceeded to running CV_m and CV_s synchronization.

Practically it works that CV_mcontinually sends (after 5 seconds interval, through JSRC) information about an actual time that displays animations and $CV_s$with respect to own actual animation time, a real measured animation speed and network communication delay adjusts speed of own animation to achieve a synchronous time.

This way of synchronization currently requires an exact time on both equipments.

7. COMPARISON WITH OTHER SOLUTION TYPES

7.1. Web vs desktop application

There are three fundamental reasons for using web browser instead of native desktop application:

- web browser is available for all ordinarily available OS without a dependency on hardware components,
- theoretically seamless network connection of logical processes in various networks,
- it is free.

The reasons why the web simulation (through WebRTC) is not an ideal way:

- offending WebRTC,
- low computing performance of JavaScript,
- a problem of a connection to yet existing architecture/product relative to an unique reports size.

7.2. Comparison with other standard solutions

As noted in chapter 2.3 relative to found problems and limitations an aim of this work is to create the environment for realization of small simulations (to 20 logical processes) not to create a competition usually by a military solution.

Only a brief look back at contemporary most used standards:

- **HLA** is a very extensive and complex solution its implementation is very demanding according to all standards. For that reason using of HLA is wholly unpractical for models with a small number of logical processes on that this work is just focused on. (Kuhl, Judith, and Weatherly 2000)
- **DIS** is in general replaced by HLA today, from this view already only a thought of network connections after this pattern (each with each) is interesting but as described in chapter 2.3 this approach is not usable.
- **TENA** is an extensive complex architecture (as HLA is) created for needs of US Army

comprising LVC distributed simulations with emphasis on live simulation thus totally exceeds scope of this work. (Powell and Noseworthy 2012)

8. TESTING A USE CASE

For testing road/highway network was chosen because it is possible to set up almost any (and still realistically looking and behaving) distributed model from several prepared types of logical processes.

These types of logical processes are used (see fig. 8):

- highway with ramp on/off (fig. 8a),
- straight stretch of highway,
- highway multilevel intersection (fig. 8c),
- road stretch with small window with refreshments (fig. 8b, primarily for testing queue realization).

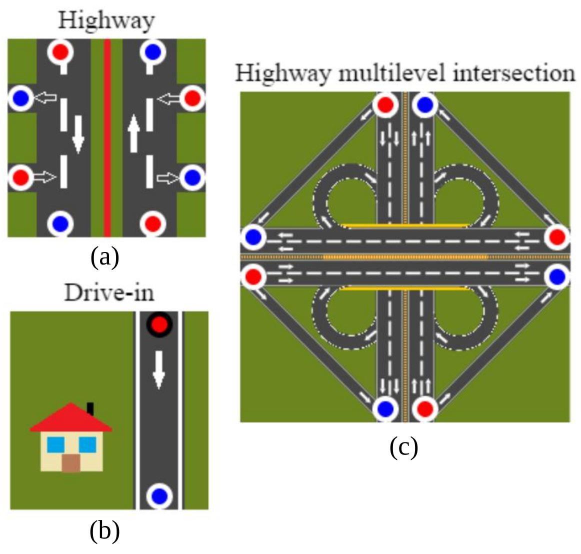


Figure 8: Examples of types of logical processes

The testing runs in local network, at 20 computers where only web browser runs with one opened panel.
It is about a medium performance ordinarily available computers with Intel® Core™ i3 processor.
Schemes of used models (topology of logical processes connections is primarily important) are in figure 9a-c. Model in figure 9c represents closed system that detains entities and serves to testing behavior of solution in extreme (i.e. normally absent) amount of entities. Then in a table 1 measured values for indexes are shown that normally guarantee smooth run of simulation. So these values indicate limits (especially table 1 / fig. 9c) of an applicability on above mentioned processor. A more powerful processor may enable you to smoothly animate a larger number of entities on other parameters will have no effect.

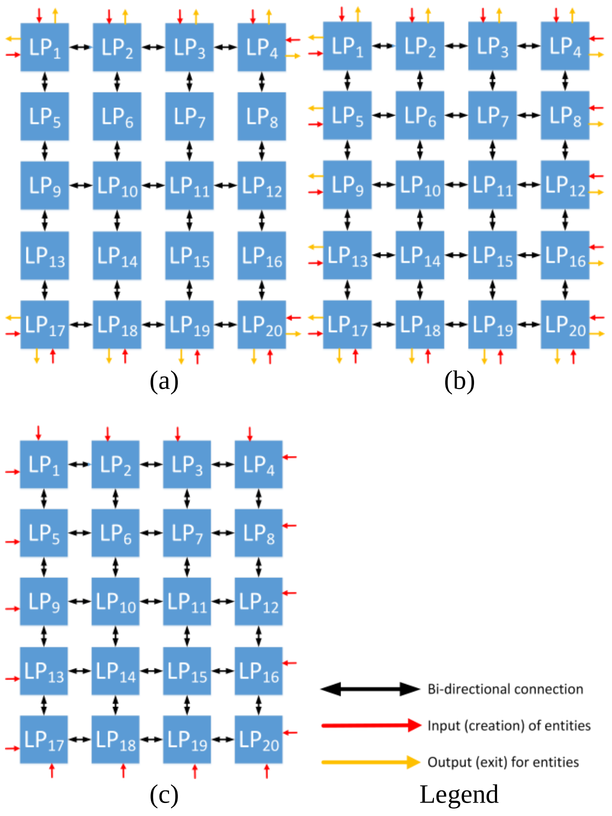

(a)
(b)
(c)
Bi-directional connection
Input (creation) of entities
Output (exit) for entities
Legend

Figure 9: Schemas of tested models

Table 1: The average values of run of distributed simulations on models by fig. 9, lookahead 500 ms

	Schemes of model in fig. 9		
Scheme	Fig. 9a	Fig. 9b	Fig. 9c
Time from start [min]	30	30	15
Count of entities in	220	450	1400
Time between request for null message and response [ms]	18,7	15,4	12,2
Time between send WebRTC message and response delivery [ms]	5,39	9,5	7,7
FPS of animation output	141	125	103
Status	No problem, values continued in time unchanged		Last smooth anim.

9. FUTURE DEVELOPMENT

Still a space remains here for future development especially in terms of solving problem with more connections and stable logical processes connection among more local networks operated behind NAT routers.
The future development will be focused too to a possibility to save the status of logical processes at server and thus allow to supervisory staff to return to

required moment whether for any simulation play and for a initiation of different solution procedure.

So option will arise to later (after finishing simulation) repeat course or to change procedure from chosen moment of time.

Hereafter there is the space for the whole series of improvements starting with improving $\mathtt{CV}_m$ and $\mathtt{CV}_s$ synchronization. Creating of animation for monitoring entity transitions among logical processes, etc.

10. CONCLUSION

Presented solution creates the environment for service training or testing whenever and wherever using only web browser that today it is available practically at any computer connected to a computer network.

Just this feature is key and competitive against existing alternatives that usually cannot do without an specific, single purpose and costly software.

However JavaScript character and WebRTC problems yet close a path to extensive use and soon we cannot expect an essential progress.

A final product can find an application for smaller models (ca around 10 logical processes) or in static distributed scenes.

REFERENCES

Fujimoto, R.M., 2000. *Parallel and distribution simulation systems*. New York: Wiley.

Kartak, S., Kavicka, A., 2014. WebRTC Technology as a Solution for a Web-Based Distributed Simulation. *Proceedings of the European Modeling and Simulation Symposium 2014,* pp. 343–349. Genova: Università di Genova.

Tropper, C., 2002. *Parallel and distributed discrete event simulation*. New York: Nova Science Pub Inc.

Kuhl, Frederick, Judith Dahmann, and Richard Weatherly. *Creating computer simulation systems : an introduction to the high level architecture.* Upper Saddle River, NJ: Prentice Hall PTR, 2000.

Bergkvist, A., Burnett, D.C., Jennings, C., Anant Narayanan, 2013. *WebRTC 1.0: Real-time Communication Between Browsers.* W3C. Available from: http://www.w3.org/TR/webrtc/ [accessed date July 2014]

Rosenberg, J., 2014. *Interactive Connectivity Establishment (ICE)*. IETF. Available from: http://tools.ietf.org/html/rfc5245 [accessed date May 2014]

Hridel, J., Kartak, S., 2013. Web-based simulation in teaching. *The European Simulation and Modelling Conference 2013*. EUROSIS-ETI.

Powell, E.T., Noseworthy, J.R., 2012. *The Test and Training Enabling Architecture (TENA).* Available from: https://www.tena-sda.org/download/attachments/6750/TENA-2012-Paper-Final.pdf [accessed date 1 July 2015]

IEEE standard for distributed interactive simulation: application protocols. New York: Institute of Electrical and Electronics Engineers, 2012.

IEEE standard for distributed interactive simulation communication services and profiles. New York: Institute of Electrical and Electronics Engineers, 1996. Print.