

Test bed for network protocols optimization

Afan Ceco, Member IEEE; Sasa Mrdovic, Senior Member IEEE

Abstract— This paper describes the test platform for verifying the functionality of network protocols and for optimization of their parameters. The test bed is made using combined OPNET simulator and MATLAB development environment. This test platform connects OPNET network protocols simulator with MATLAB development environment in the way that OPNET runs simulations of network traffic, with the predetermined parameter values, while MATLAB executes the script with a mathematical algorithm, which optimizes parameters listed in OPNET simulator.

Keywords— test platform, OPNET, MATLAB, network protocols, optimization

I. INTRODUCTION

It is very often necessary to test network protocols in the simulation environment. The reason why it is necessary is the impossibility of testing in real networks, so as not to disturb the real traffic. Network simulators (e.g. OPNET) have no support for non-standard parameters or their dynamic change. On the other hand, MATLAB is suitable for testing ideas and concepts, because it represents a high level programming language and a development environment. Enabling these two components connection together provides the best of both worlds.

The network simulators such as NS-2 [1], the NS-3 [2], OPNET [3], OMNET++ [4] or others, allow checking the validity of networking concepts, as well as changes in the network protocols. However, they lack the power of specialized software for mathematical calculations, such as MATLAB [5], when it comes to complex mathematical operations (optimization, 3-D models, etc.). In this paper, a test platform is presented, which can be used to verify the functionality of the new network protocol, that would include optimization techniques, or verification of the existing protocols, in addition to optimization of their parameters. It

consists of a connected simulator OPNET Modeler 14.5 and a development environment MATLAB R2012.

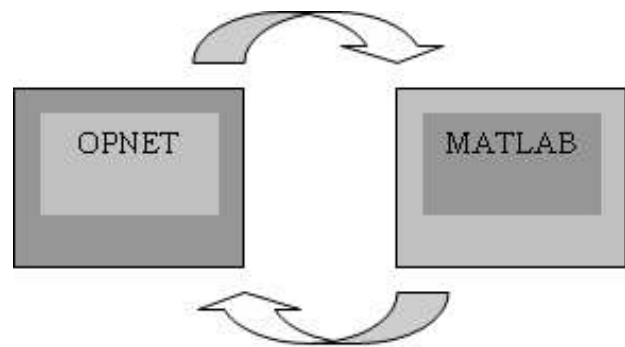


Fig. 1. Link between OPNET and MATLAB

This test platform connects OPNET simulator of the network protocols with MATLAB development environment in the way that OPNET runs simulations of network traffic, with the predetermined parameter values, while MATLAB executes the script with a mathematical algorithm, which optimizes the parameters listed in OPNET simulator.

The first part of this paper presents the situation in the subject area. The second part describes OPNET simulator. In the third part, MATLAB development environment is presented. In the fourth part, the test platform is explained, consisting of a paired OPNET and MATLAB, and its implementation. At the end of this paper, the results of simulations are presented, obtained by optimizing the parameters, and executed on the subject test platform.

II. REVIEW OF THE SUBJECT AREA

In the research papers which can be found in this area, the models for linking different simulators are described, as well as for linking simulators and other software tools. Very often,

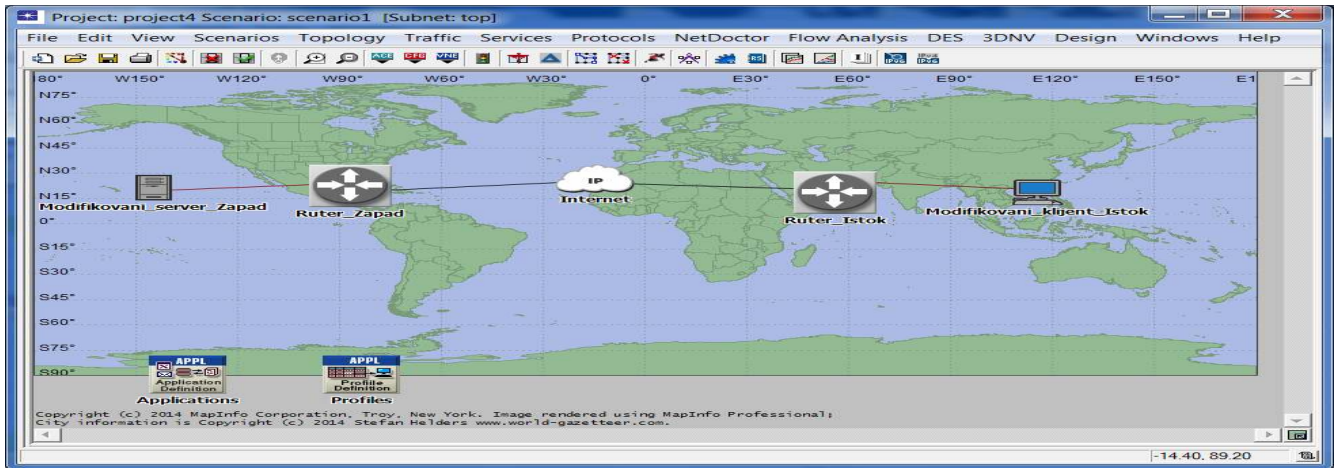


Fig. 2. OPNET Simulator – Test network topology

it is the case of a so-called co-simulation, which represents a problem division into two or more parts, which are simultaneously simulated on a variety of simulators. The second most common use is to connect two simulators to validate the results from one simulator, by checking them on the second simulator (from another manufacturer). Co-simulation, as well as the use of two different simulators together, is the topic of Paper [6]. The authors describe a platform for co-simulation, which consists of connected simulators OPNET and SIMULINK. SIMULINK environment for simulation is based on block diagrams and on design of models [7]. Both simulators are executed in parallel, and interact with the synchronization, whereby the main simulator is OPNET. In Paper [8], the authors present the implementation of connectivity of OMNET++ simulator and MATLAB development environment, with an emphasis on co-simulation. Paper [9] presents an example of connecting OPNET and MATLAB, provided that it is done for the purpose of validation on MATLAB of the results obtained on OPNET simulator.

Unlike the above described models of linking the simulators, we herein present the connection of OPNET and MATLAB in a common test platform, where they act as a single combined simulator, which utilizes the advantages of both software environments. This test platform uses an iterative principle in order to reach the best possible results, and through a number of (automated) starting and running of simulations, it leads to the optimization of the simulation parameters.

III. OPNET SIMULATOR

OPNET is a simulator of network protocols and communications, a product of the company named Riverbed, and there are several different variants of this product. A very popular variant is called “OPNET IT Guru Academic Edition”, especially in the academic community, because it was free for educational use. However, in this version,

simulator users were not allowed to modify the source code of the simulator. At present, the current replacement for this variant is called “OPNET Modeler Academic Edition”, and was presented in 2014. In this paper, we used a commercial version of OPNET Modeler, mostly because this version of the simulator has the option of changing the source code. This allows modification of the complete behavior of network protocols being a part of the simulator. Figure 2 shows a GUI of OPNET simulator with a built topology network that is used to test the proposed linking of OPNET and MATLAB. The topology consists of a server, which is connected via router to the internet network, then the internet network itself (Internet), and a client, who is also connected via router to the internet network.

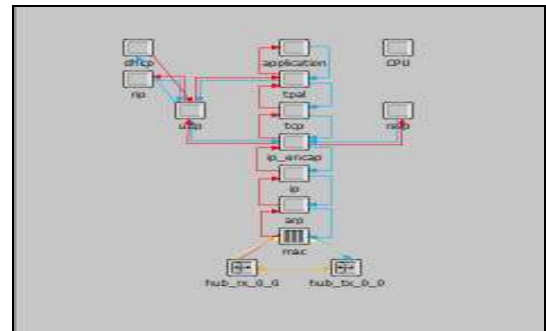


Fig. 3. OPNET Simulator – Display of protocols for client node

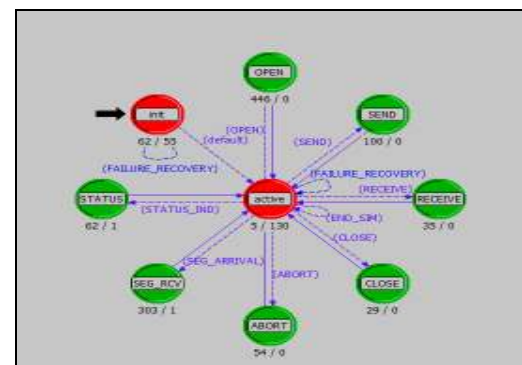


Fig. 4. OPNET Simulator – State diagram for TCP protocol

Figure 3 presents the protocols for the client node, i.e. all protocols that are available on the simulator for that node and that can be changed. Figure 4 shows the state diagram for one of the protocols on the client node, and in this case it is the TCP protocol. It is important to point out the fact that each of the states in the diagram can be opened, with its code in C/C++ displayed, which can be changed, if necessary.

Figures 5 and 6 show the parameters of the TCP protocol, as well as the function block with the C/C++ code that can be changed. The parameter values in Fig. 5 can be changed programmatically by changing the code as shown in Fig. 6. Once we perform the change in the code of the simulator, it is necessary to recompile the simulator.

Attribute Name	Type	Units	Primary Key	Default Value	Prominent	Tags
Version/Flavor	string			Unspecified		
Maximum Segment Size	integer	bytes		Auto-Assigned		
Receive Buffer	integer	bytes		8760		
Receive Buffer Adjustment	integer			None		
Receive Buffer Usage Threshold	double	of RCV BUFF		0.0		
Delayed ACK Mechanism	integer			Segment/Clock Based		
Maximum ACK Delay	double	sec		0.200		
Maximum ACK Segments	integer			2		
Slow-Start Initial Count	integer	MSS		2		
Fast Retransmit	toggle			Disabled		
Duplicate ACK Threshold	integer			3		

Fig. 5. OPNET Simulator – Display of parameters

```

1 static void
2 tcp_mgr_initialize(void)
3 {
4     /* initialize the static variables used in this module. */
5     tcp_mgr_initialize();
6
7     /* initialize the variables used for process registry. */
8     tcp_mgr_initialize();
9
10    /* initialize the module's object. */
11    tcp_mgr_initialize();
12
13    /* initialize the top manager's system's parameters. */
14    tcp_mgr_initialize();
15
16    /* initialize the name of the process, or in the "process model" attribute of the module. */
17    tcp_mgr_initialize();
18
19    /* initialize the diagram's structure dynamic array pointer. */
20    tcp_mgr_initialize();
21
22    /* other stuff. */
23    tcp_mgr_initialize();
24
25    tcp_mgr_initialize();
26
27    /* initialize the module's object. */
28    tcp_mgr_initialize();
29
30    /* initialize the module's object. */
31    tcp_mgr_initialize();
32
33    /* initialize the module's object. */
34    tcp_mgr_initialize();
35
36    /* initialize the module's object. */
37    tcp_mgr_initialize();
38
39    /* initialize the module's object. */
40    tcp_mgr_initialize();
41
42    /* initialize the module's object. */
43    tcp_mgr_initialize();
44
45    /* initialize the module's object. */
46    tcp_mgr_initialize();
47
48    /* initialize the module's object. */
49    tcp_mgr_initialize();
50
51    /* initialize the module's object. */
52    tcp_mgr_initialize();
53
54    /* initialize the module's object. */
55    tcp_mgr_initialize();
56
57    /* initialize the module's object. */
58    tcp_mgr_initialize();
59
60    /* initialize the module's object. */
61    tcp_mgr_initialize();
62
63    /* initialize the module's object. */
64    tcp_mgr_initialize();
65
66    /* initialize the module's object. */
67    tcp_mgr_initialize();
68
69    /* initialize the module's object. */
70    tcp_mgr_initialize();
71
72    /* initialize the module's object. */
73    tcp_mgr_initialize();
74
75    /* initialize the module's object. */
76    tcp_mgr_initialize();
77
78    /* initialize the module's object. */
79    tcp_mgr_initialize();
80
81    /* initialize the module's object. */
82    tcp_mgr_initialize();
83
84    /* initialize the module's object. */
85    tcp_mgr_initialize();
86
87    /* initialize the module's object. */
88    tcp_mgr_initialize();
89
90    /* initialize the module's object. */
91    tcp_mgr_initialize();
92
93    /* initialize the module's object. */
94    tcp_mgr_initialize();
95
96    /* initialize the module's object. */
97    tcp_mgr_initialize();
98
99    /* initialize the module's object. */
100   tcp_mgr_initialize();
101
102   /* initialize the module's object. */
103   tcp_mgr_initialize();
104
105   /* initialize the module's object. */
106   tcp_mgr_initialize();
107
108   /* initialize the module's object. */
109   tcp_mgr_initialize();
110
111   /* initialize the module's object. */
112   tcp_mgr_initialize();
113
114   /* initialize the module's object. */
115   tcp_mgr_initialize();
116
117   /* initialize the module's object. */
118   tcp_mgr_initialize();
119
120   /* initialize the module's object. */
121   tcp_mgr_initialize();
122
123   /* initialize the module's object. */
124   tcp_mgr_initialize();
125
126   /* initialize the module's object. */
127   tcp_mgr_initialize();
128
129   /* initialize the module's object. */
130   tcp_mgr_initialize();
131
132   /* initialize the module's object. */
133   tcp_mgr_initialize();
134
135   /* initialize the module's object. */
136   tcp_mgr_initialize();
137
138   /* initialize the module's object. */
139   tcp_mgr_initialize();
140
141   /* initialize the module's object. */
142   tcp_mgr_initialize();
143
144   /* initialize the module's object. */
145   tcp_mgr_initialize();
146
147   /* initialize the module's object. */
148   tcp_mgr_initialize();
149
150   /* initialize the module's object. */
151   tcp_mgr_initialize();
152
153   /* initialize the module's object. */
154   tcp_mgr_initialize();
155
156   /* initialize the module's object. */
157   tcp_mgr_initialize();
158
159   /* initialize the module's object. */
160   tcp_mgr_initialize();
161
162   /* initialize the module's object. */
163   tcp_mgr_initialize();
164
165   /* initialize the module's object. */
166   tcp_mgr_initialize();
167
168   /* initialize the module's object. */
169   tcp_mgr_initialize();
170
171   /* initialize the module's object. */
172   tcp_mgr_initialize();
173
174   /* initialize the module's object. */
175   tcp_mgr_initialize();
176
177   /* initialize the module's object. */
178   tcp_mgr_initialize();
179
180   /* initialize the module's object. */
181   tcp_mgr_initialize();
182
183   /* initialize the module's object. */
184   tcp_mgr_initialize();
185
186   /* initialize the module's object. */
187   tcp_mgr_initialize();
188
189   /* initialize the module's object. */
190   tcp_mgr_initialize();
191
192   /* initialize the module's object. */
193   tcp_mgr_initialize();
194
195   /* initialize the module's object. */
196   tcp_mgr_initialize();
197
198   /* initialize the module's object. */
199   tcp_mgr_initialize();
200
201   /* initialize the module's object. */
202   tcp_mgr_initialize();
203
204   /* initialize the module's object. */
205   tcp_mgr_initialize();
206
207   /* initialize the module's object. */
208   tcp_mgr_initialize();
209
210   /* initialize the module's object. */
211   tcp_mgr_initialize();
212
213   /* initialize the module's object. */
214   tcp_mgr_initialize();
215
216   /* initialize the module's object. */
217   tcp_mgr_initialize();
218
219   /* initialize the module's object. */
220   tcp_mgr_initialize();
221
222   /* initialize the module's object. */
223   tcp_mgr_initialize();
224
225   /* initialize the module's object. */
226   tcp_mgr_initialize();
227
228   /* initialize the module's object. */
229   tcp_mgr_initialize();
230
231   /* initialize the module's object. */
232   tcp_mgr_initialize();
233
234   /* initialize the module's object. */
235   tcp_mgr_initialize();
236
237   /* initialize the module's object. */
238   tcp_mgr_initialize();
239
240   /* initialize the module's object. */
241   tcp_mgr_initialize();
242
243   /* initialize the module's object. */
244   tcp_mgr_initialize();
245
246   /* initialize the module's object. */
247   tcp_mgr_initialize();
248
249   /* initialize the module's object. */
250   tcp_mgr_initialize();
251
252   /* initialize the module's object. */
253   tcp_mgr_initialize();
254
255   /* initialize the module's object. */
256   tcp_mgr_initialize();
257
258   /* initialize the module's object. */
259   tcp_mgr_initialize();
260
261   /* initialize the module's object. */
262   tcp_mgr_initialize();
263
264   /* initialize the module's object. */
265   tcp_mgr_initialize();
266
267   /* initialize the module's object. */
268   tcp_mgr_initialize();
269
270   /* initialize the module's object. */
271   tcp_mgr_initialize();
272
273   /* initialize the module's object. */
274   tcp_mgr_initialize();
275
276   /* initialize the module's object. */
277   tcp_mgr_initialize();
278
279   /* initialize the module's object. */
280   tcp_mgr_initialize();
281
282   /* initialize the module's object. */
283   tcp_mgr_initialize();
284
285   /* initialize the module's object. */
286   tcp_mgr_initialize();
287
288   /* initialize the module's object. */
289   tcp_mgr_initialize();
290
291   /* initialize the module's object. */
292   tcp_mgr_initialize();
293
294   /* initialize the module's object. */
295   tcp_mgr_initialize();
296
297   /* initialize the module's object. */
298   tcp_mgr_initialize();
299
300   /* initialize the module's object. */
301   tcp_mgr_initialize();
302
303   /* initialize the module's object. */
304   tcp_mgr_initialize();
305
306   /* initialize the module's object. */
307   tcp_mgr_initialize();
308
309   /* initialize the module's object. */
310   tcp_mgr_initialize();
311
312   /* initialize the module's object. */
313   tcp_mgr_initialize();
314
315   /* initialize the module's object. */
316   tcp_mgr_initialize();
317
318   /* initialize the module's object. */
319   tcp_mgr_initialize();
320
321   /* initialize the module's object. */
322   tcp_mgr_initialize();
323
324   /* initialize the module's object. */
325   tcp_mgr_initialize();
326
327   /* initialize the module's object. */
328   tcp_mgr_initialize();
329
330   /* initialize the module's object. */
331   tcp_mgr_initialize();
332
333   /* initialize the module's object. */
334   tcp_mgr_initialize();
335
336   /* initialize the module's object. */
337   tcp_mgr_initialize();
338
339   /* initialize the module's object. */
340   tcp_mgr_initialize();
341
342   /* initialize the module's object. */
343   tcp_mgr_initialize();
344
345   /* initialize the module's object. */
346   tcp_mgr_initialize();
347
348   /* initialize the module's object. */
349   tcp_mgr_initialize();
350
351   /* initialize the module's object. */
352   tcp_mgr_initialize();
353
354   /* initialize the module's object. */
355   tcp_mgr_initialize();
356
357   /* initialize the module's object. */
358   tcp_mgr_initialize();
359
360   /* initialize the module's object. */
361   tcp_mgr_initialize();
362
363   /* initialize the module's object. */
364   tcp_mgr_initialize();
365
366   /* initialize the module's object. */
367   tcp_mgr_initialize();
368
369   /* initialize the module's object. */
370   tcp_mgr_initialize();
371
372   /* initialize the module's object. */
373   tcp_mgr_initialize();
374
375   /* initialize the module's object. */
376   tcp_mgr_initialize();
377
378   /* initialize the module's object. */
379   tcp_mgr_initialize();
380
381   /* initialize the module's object. */
382   tcp_mgr_initialize();
383
384   /* initialize the module's object. */
385   tcp_mgr_initialize();
386
387   /* initialize the module's object. */
388   tcp_mgr_initialize();
389
390   /* initialize the module's object. */
391   tcp_mgr_initialize();
392
393   /* initialize the module's object. */
394   tcp_mgr_initialize();
395
396   /* initialize the module's object. */
397   tcp_mgr_initialize();
398
399   /* initialize the module's object. */
400   tcp_mgr_initialize();
401
402   /* initialize the module's object. */
403   tcp_mgr_initialize();
404
405   /* initialize the module's object. */
406   tcp_mgr_initialize();
407
408   /* initialize the module's object. */
409   tcp_mgr_initialize();
410
411   /* initialize the module's object. */
412   tcp_mgr_initialize();
413
414   /* initialize the module's object. */
415   tcp_mgr_initialize();
416
417   /* initialize the module's object. */
418   tcp_mgr_initialize();
419
420   /* initialize the module's object. */
421   tcp_mgr_initialize();
422
423   /* initialize the module's object. */
424   tcp_mgr_initialize();
425
426   /* initialize the module's object. */
427   tcp_mgr_initialize();
428
429   /* initialize the module's object. */
430   tcp_mgr_initialize();
431
432   /* initialize the module's object. */
433   tcp_mgr_initialize();
434
435   /* initialize the module's object. */
436   tcp_mgr_initialize();
437
438   /* initialize the module's object. */
439   tcp_mgr_initialize();
440
441   /* initialize the module's object. */
442   tcp_mgr_initialize();
443
444   /* initialize the module's object. */
445   tcp_mgr_initialize();
446
447   /* initialize the module's object. */
448   tcp_mgr_initialize();
449
450   /* initialize the module's object. */
451   tcp_mgr_initialize();
452
453   /* initialize the module's object. */
454   tcp_mgr_initialize();
455
456   /* initialize the module's object. */
457   tcp_mgr_initialize();
458
459   /* initialize the module's object. */
460   tcp_mgr_initialize();
461
462   /* initialize the module's object. */
463   tcp_mgr_initialize();
464
465   /* initialize the module's object. */
466   tcp_mgr_initialize();
467
468   /* initialize the module's object. */
469   tcp_mgr_initialize();
470
471   /* initialize the module's object. */
472   tcp_mgr_initialize();
473
474   /* initialize the module's object. */
475   tcp_mgr_initialize();
476
477   /* initialize the module's object. */
478   tcp_mgr_initialize();
479
480   /* initialize the module's object. */
481   tcp_mgr_initialize();
482
483   /* initialize the module's object. */
484   tcp_mgr_initialize();
485
486   /* initialize the module's object. */
487   tcp_mgr_initialize();
488
489   /* initialize the module's object. */
490   tcp_mgr_initialize();
491
492   /* initialize the module's object. */
493   tcp_mgr_initialize();
494
495   /* initialize the module's object. */
496   tcp_mgr_initialize();
497
498   /* initialize the module's object. */
499   tcp_mgr_initialize();
500
501   /* initialize the module's object. */
502   tcp_mgr_initialize();
503
504   /* initialize the module's object. */
505   tcp_mgr_initialize();
506
507   /* initialize the module's object. */
508   tcp_mgr_initialize();
509
510   /* initialize the module's object. */
511   tcp_mgr_initialize();
512
513   /* initialize the module's object. */
514   tcp_mgr_initialize();
515
516   /* initialize the module's object. */
517   tcp_mgr_initialize();
518
519   /* initialize the module's object. */
520   tcp_mgr_initialize();
521
522   /* initialize the module's object. */
523   tcp_mgr_initialize();
524
525   /* initialize the module's object. */
526   tcp_mgr_initialize();
527
528   /* initialize the module's object. */
529   tcp_mgr_initialize();
530
531   /* initialize the module's object. */
532   tcp_mgr_initialize();
533
534   /* initialize the module's object. */
535   tcp_mgr_initialize();
536
537   /* initialize the module's object. */
538   tcp_mgr_initialize();
539
540   /* initialize the module's object. */
541   tcp_mgr_initialize();
542
543   /* initialize the module's object. */
544   tcp_mgr_initialize();
545
546   /* initialize the module's object. */
547   tcp_mgr_initialize();
548
549   /* initialize the module's object. */
550   tcp_mgr_initialize();
551
552   /* initialize the module's object. */
553   tcp_mgr_initialize();
554
555   /* initialize the module's object. */
556   tcp_mgr_initialize();
557
558   /* initialize the module's object. */
559   tcp_mgr_initialize();
560
561   /* initialize the module's object. */
562   tcp_mgr_initialize();
563
564   /* initialize the module's object. */
565   tcp_mgr_initialize();
566
567   /* initialize the module's object. */
568   tcp_mgr_initialize();
569
570   /* initialize the module's object. */
571   tcp_mgr_initialize();
572
573   /* initialize the module's object. */
574   tcp_mgr_initialize();
575
576   /* initialize the module's object. */
577   tcp_mgr_initialize();
578
579   /* initialize the module's object. */
580   tcp_mgr_initialize();
581
582   /* initialize the module's object. */
583   tcp_mgr_initialize();
584
585   /* initialize the module's object. */
586   tcp_mgr_initialize();
587
588   /* initialize the module's object. */
589   tcp_mgr_initialize();
590
591   /* initialize the module's object. */
592   tcp_mgr_initialize();
593
594   /* initialize the module's object. */
595   tcp_mgr_initialize();
596
597   /* initialize the module's object. */
598   tcp_mgr_initialize();
599
600   /* initialize the module's object. */
601   tcp_mgr_initialize();
602
603   /* initialize the module's object. */
604   tcp_mgr_initialize();
605
606   /* initialize the module's object. */
607   tcp_mgr_initialize();
608
609   /* initialize the module's object. */
610   tcp_mgr_initialize();
611
612   /* initialize the module's object. */
613   tcp_mgr_initialize();
614
615   /* initialize the module's object. */
616   tcp_mgr_initialize();
617
618   /* initialize the module's object. */
619   tcp_mgr_initialize();
620
621   /* initialize the module's object. */
622   tcp_mgr_initialize();
623
624   /* initialize the module's object. */
625   tcp_mgr_initialize();
626
627   /* initialize the module's object. */
628   tcp_mgr_initialize();
629
630   /* initialize the module's object. */
631   tcp_mgr_initialize();
632
633   /* initialize the module's object. */
634   tcp_mgr_initialize();
635
636   /* initialize the module's object. */
637   tcp_mgr_initialize();
638
639   /* initialize the module's object. */
640   tcp_mgr_initialize();
641
642   /* initialize the module's object. */
643   tcp_mgr_initialize();
644
645   /* initialize the module's object. */
646   tcp_mgr_initialize();
647
648   /* initialize the module's object. */
649   tcp_mgr_initialize();
650
651   /* initialize the module's object. */
652   tcp_mgr_initialize();
653
654   /* initialize the module's object. */
655   tcp_mgr_initialize();
656
657   /* initialize the module's object. */
658   tcp_mgr_initialize();
659
660   /* initialize the module's object. */
661   tcp_mgr_initialize();
662
663   /* initialize the module's object. */
664   tcp_mgr_initialize();
665
666   /* initialize the module's object. */
667   tcp_mgr_initialize();
668
669   /* initialize the module's object. */
670   tcp_mgr_initialize();
671
672   /* initialize the module's object. */
673   tcp_mgr_initialize();
674
675   /* initialize the module's object. */
676   tcp_mgr_initialize();
677
678   /* initialize the module's object. */
679   tcp_mgr_initialize();
680
681   /* initialize the module's object. */
682   tcp_mgr_initialize();
683
684   /* initialize the module's object. */
685   tcp_mgr_initialize();
686
687   /* initialize the module's object. */
688   tcp_mgr_initialize();
689
690   /* initialize the module's object. */
691   tcp_mgr_initialize();
692
693   /* initialize the module's object. */
694   tcp_mgr_initialize();
695
696   /* initialize the module's object. */
697   tcp_mgr_initialize();
698
699   /* initialize the module's object. */
700   tcp_mgr_initialize();
701
702   /* initialize the module's object. */
703   tcp_mgr_initialize();
704
705   /* initialize the module's object. */
706   tcp_mgr_initialize();
707
708   /* initialize the module's object. */
709   tcp_mgr_initialize();
710
711   /* initialize the module's object. */
712   tcp_mgr_initialize();
713
714   /* initialize the module's object. */
715   tcp_mgr_initialize();
716
717   /* initialize the module's object. */
718   tcp_mgr_initialize();
719
720   /* initialize the module's object. */
721   tcp_mgr_initialize();
722
723   /* initialize the module's object. */
724   tcp_mgr_initialize();
725
726   /* initialize the module's object. */
727   tcp_mgr_initialize();
728
729   /* initialize the module's object. */
730   tcp_mgr_initialize();
731
732   /* initialize the module's object. */
733   tcp_mgr_initialize();
734
735   /* initialize the module's object. */
736   tcp_mgr_initialize();
737
738   /* initialize the module's object. */
739   tcp_mgr_initialize();
740
741   /* initialize the module's object. */
742   tcp_mgr_initialize();
743
744   /* initialize the module's object. */
745   tcp_mgr_initialize();
746
747   /* initialize the module's object. */
748   tcp_mgr_initialize();
749
750   /* initialize the module's object. */
751   tcp_mgr_initialize();
752
753   /* initialize the module's object. */
754   tcp_mgr_initialize();
755
756   /* initialize the module's object. */
757   tcp_mgr_initialize();
758
759   /* initialize the module's object. */
760   tcp_mgr_initialize();
761
762   /* initialize the module's object. */
763   tcp_mgr_initialize();
764
765   /* initialize the module's object. */
766   tcp_mgr_initialize();
767
768   /* initialize the module's object. */
769   tcp_mgr_initialize();
770
771   /* initialize the module's object. */
772   tcp_mgr_initialize();
773
774   /* initialize the module's object. */
775   tcp_mgr_initialize();
776
777   /* initialize the module's object. */
778   tcp_mgr_initialize();
779
780   /* initialize the module's object. */
781   tcp_mgr_initialize();
782
783   /* initialize the module's object. */
784   tcp_mgr_initialize();
785
786   /* initialize the module's object. */
787   tcp_mgr_initialize();
788
789   /* initialize the module's object. */
790   tcp_mgr_initialize();
791
792   /* initialize the module's object. */
793   tcp_mgr_initialize();
794
795   /* initialize the module's object. */
796   tcp_mgr_initialize();
797
798   /* initialize the module's object. */
799   tcp_mgr_initialize();
800
801   /* initialize the module's object. */
802   tcp_mgr_initialize();
803
804   /* initialize the module's object. */
805   tcp_mgr_initialize();
806
807   /* initialize the module's object. */
808   tcp_mgr_initialize();
809
810   /* initialize the module's object. */
811   tcp_mgr_initialize();
812
813   /* initialize the module's object. */
814   tcp_mgr_initialize();
815
816   /* initialize the module's object. */
817   tcp_mgr_initialize();
818
819   /* initialize the module's object. */
820   tcp_mgr_initialize();
821
822   /* initialize the module's object. */
823   tcp_mgr_initialize();
824
825   /* initialize the module's object. */
826   tcp_mgr_initialize();
827
828   /* initialize the module's object. */
829   tcp_mgr_initialize();
830
831   /* initialize the module's object. */
832   tcp_mgr_initialize();
833
834   /* initialize the module's object. */
835   tcp_mgr_initialize();
836
837   /* initialize the module's object. */
838   tcp_mgr_initialize();
839
840   /* initialize the module's object. */
841   tcp_mgr_initialize();
842
843   /* initialize the module's object. */
844   tcp_mgr_initialize();
845
846   /* initialize the module's object. */
847   tcp_mgr_initialize();
848
849   /* initialize the module's object. */
850   tcp_mgr_initialize();
851
852   /* initialize the module's object. */
853   tcp_mgr_initialize();
854
855   /* initialize the module's object. */
856   tcp_mgr_initialize();
857
858   /* initialize the module's object. */
859   tcp_mgr_initialize();
860
861   /* initialize the module's object. */
862   tcp_mgr_initialize();
863
864   /* initialize the module's object. */
865   tcp_mgr_initialize();
866
867   /* initialize the module's object. */
868   tcp_mgr_initialize();
869
870   /* initialize the module's object. */
871   tcp_mgr_initialize();
872
873   /* initialize the module's object. */
874   tcp_mgr_initialize();
875
876   /* initialize the module's object. */
877   tcp_mgr_initialize();
878
879   /* initialize the module's object. */
880   tcp_mgr_initialize();
881
882   /* initialize the module's object. */
883   tcp_mgr_initialize();
884
885   /* initialize the module's object. */
886   tcp_mgr_initialize();
887
888   /* initialize the module's object. */
889   tcp_mgr_initialize();
890
891   /* initialize the module's object. */
892   tcp_mgr_initialize();
893
894   /* initialize the module's object. */
895   tcp_mgr_initialize();
896
897   /* initialize the module's object. */
898   tcp_mgr_initialize();
899
900   /* initialize the module's object. */
901   tcp_mgr_initialize();
902
903   /* initialize the module's object. */
904   tcp_mgr_initialize();
905
906   /* initialize the module's object. */
907   tcp_mgr_initialize();
908
909   /* initialize the module's object. */
910   tcp_mgr_initialize();
911
912   /* initialize the module's object. */
913   tcp_mgr_initialize();
914
915   /* initialize the module's object. */
916   tcp_mgr_initialize();
917
918   /* initialize the module's object. */
919   tcp_mgr_initialize();
920
921   /* initialize the module's object. */
922   tcp_mgr_initialize();
923
924   /* initialize the module's object. */
925   tcp_mgr_initialize();
926
927   /* initialize the module's object. */
928   tcp_mgr_initialize();
929
930   /* initialize the module's object. */
931   tcp_mgr_initialize();
932
933   /* initialize the module's object. */
934   tcp_mgr_initialize();
935
936   /* initialize the module's object. */
937   tcp_mgr_initialize();
938
939   /* initialize the module's object. */
940   tcp_mgr_initialize();
941
942   /* initialize the module's object. */
943   tcp_mgr_initialize();
944
945   /* initialize the module's object. */
946   tcp_mgr_initialize();
947
948   /* initialize the module's object. */
949   tcp_mgr_initialize();
950
951   /* initialize the module's object. */
952   tcp_mgr_initialize();
953
954   /* initialize the module's object. */
955   tcp_mgr_initialize();
956
957   /* initialize the module's object. */
958   tcp_mgr_initialize();
959
960   /* initialize the module's object. */
961   tcp_mgr_initialize();
962
963   /* initialize the module's object. */
964   tcp_mgr_initialize();
965
966   /* initialize the module's object. */
967   tcp_mgr_initialize();
968
969   /* initialize the module's object. */
970   tcp_mgr_initialize();
971
972   /* initialize the module's object. */
973   tcp_mgr_initialize();
974
975   /* initialize the module's object. */
976   tcp_mgr_initialize();
977
978   /* initialize the module's object. */
979   tcp_mgr_initialize();
980
981   /* initialize the module's object. */
982   tcp_mgr_initialize();
983
984   /* initialize the module's object. */
985   tcp_mgr_initialize();
986
987   /* initialize the module's object. */
988   tcp_mgr_initialize();
989
990   /* initialize the module's object. */
991   tcp_mgr_initialize();
992
993   /* initialize the module's object. */
994   tcp_mgr_initialize();
995
996   /* initialize the module's object. */
997   tcp_mgr_initialize();
998
999   /* initialize the module's object. */
1000  tcp_mgr_initialize();

```

Fig. 6. OPNET Simulator – Function block

In the presented case, we have used MS Visual C++ 9.0 (MS Visual Studio 2008) on MS Windows 7. It is necessary to set the environment variables for the compiler and simulator to be bound. All the details related to the changing code of OPNET Modeler are explained in Book [10]. It should be noted that the operating system, and simulator, and compiler, should be from the same generation, so that they appeared on the market in about the same time, because otherwise some compatibility issues arise.

By changing the code in OPNET simulator, the existing protocol can be changed to work differently, or we can even create an entirely new network protocol. The precondition is,

of course, that we have installed C/C++ compiler, and that the environment variables are set.

IV. MATLAB DEVELOPMENT ENVIRONMENT

MATLAB is a development environment and a high level programming language for numerical and matrix computation. It is a product of the company named MathWorks, which offers a SIMULINK simulator as well.

MATLAB represents all data in the form of arrays. The routines serving to manipulate the arrays in MATLAB are starting with the prefix mx. Import and export of data to and from MATLAB can be made in several ways. Below are presented some of them.

MATLAB can invoke functions and procedures written in the C programming language or in Fortran. In this, the so-called MEX files are commonly used, which represent the interface between MATLAB and subroutines written in C, C++ or Fortran. When compiled, MEX files are dynamically loaded and allow other code to be called within MATLAB, as if it were its own functions. With a MEX file, of course, we can read or write data. A detailed explanation can be found in the manual issued by the MATLAB development environment manufacturer [11].

In contrast, the so-called MAT files allow the import and export of data to or from MATLAB environment in a direct way. MAT files are actually files that MATLAB uses to store data on the disk. MAT files provide an easy and convenient mechanism for transporting data between different platforms. Also, they allow the import and export of data into MATLAB from some other MATLAB or stand-alone applications. In order to simplify the use of MAT files from other applications, a library of access routines is offered, and they all begin with the prefix mat, which makes reading and writing of MAT files very easy using C or Fortran program [11].

It is possible to load data into MATLAB also from ASCII files. Such files must have data written in ASCII format, with rows of fixed length ending with the transition to a new line, as well as spaces that separate the numbers. Data is retrieved from this file by using the load command, and are recorded using the save command [11].

V. COUPLING OF OPNET AND MATLAB

In this paper, we present a test platform that connects OPNET simulator of network protocols with MATLAB development environment. OPNET runs simulations of network traffic, with the predetermined values of parameters, such as the percentage of a packet loss, and other parameters of the TCP connection. Upon completion of the simulation, the values of the variables affecting the parameters we have been looking to improve are recorded, and the script with the

mathematical algorithm, which optimizes the above parameters, is executed in MATLAB, for the purposes of OPNET simulator. In Fig. 7, a flow diagram of the execution of the simulation and optimization is shown, which can be applied to any communication protocol. The initial parameter values (x_1, y_1) are used in OPNET simulator as input parameters. After the simulation ends, the result obtained is shown on the diagram as the value (R). This value (R) is imported into MATLAB in order to carry out the effectiveness check of the initial parameters values. Subsequently, the optimization algorithm in MATLAB is changing the initial parameters (x_1, y_1) to (x_2, y_2), which is presented in Fig. 7 by operator λ . These parameters (x_2, y_2) are recorded now as the initial parameters (x_1, y_1). During the next start-up of simulation in OPNET, the new input parameters are being loaded. The result (R) is returned back to MATLAB and is compared with the previous result. If the result is better, the optimization algorithm continues in this direction with the further modification of parameters. If the result is worse, the optimization algorithm rejects the previously caused modification. The link between OPNET and MATLAB can be accomplished in several ways, and these methods can be divided into two groups: using files or software methods for direct applications linking. In this case, the connection is achieved using ASCII files and by recording the variables in the file from one software, and then loading the same variables from the given file in another software.

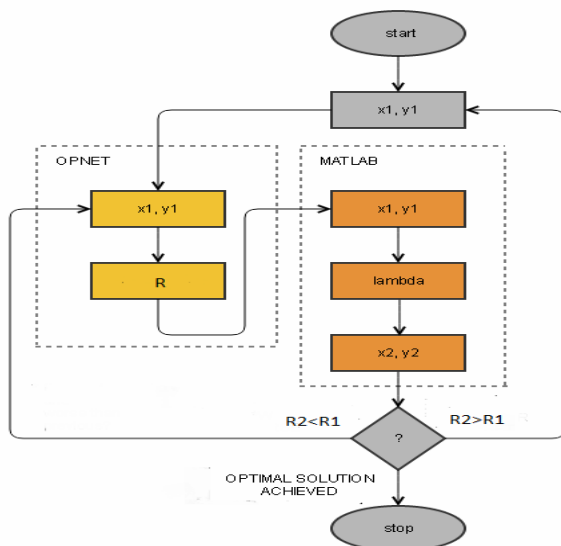


Fig. 7. Flow Diagram – Execution of simulation with the relationship between OPNET and MATLAB

The connection between MATLAB and OPNET can be achieved without the files, using COM (Component Object Model) software components, or a similar method of

communication between processes, or for the connection of various applications (such as OLE, ActiveX, etc., all of which are derived from a COM model). Starting OPNET simulation large number of times, along with the execution of MATLAB scripts each time, is provided by the software for actions automation on computer system called "MurGee" [12]. OPNET simulator has also its own options for automating the execution of simulations, except that in this case we should ensure the calls of MATLAB scripts, in each iteration cycle.

VII. CONCLUSION

This paper shows an example of connecting OPNET simulator and MATLAB development environment in a common test platform, using files as a means of communication between these two software environments. The presented test platform can be used to verify the functionality of the new network protocol that would include optimization techniques, or verification of the existing protocols in addition to optimization of their parameters. OPNET runs simulations of network traffic, with the predetermined parameter values, while MATLAB executes the script with the mathematical algorithm, which optimizes the parameters in OPNET simulator. Operation of this test platform has been tested on the example of optimization of parameters for the TCP protocol, which is shown in the form of results, with the performance obtained by simulation, before and after the application of optimization.

REFERENCES

- [1] NS-2 URL: http://nslam.isi.edu/nslam/index.php/User_Information
- [2] NS-3 URL: <http://www.nslam.org/>
- [3] Riverbed OPNET Modeler URL: <http://www.riverbed.com/products-solutions/products/networkplanning-simulation/Network-Simulation.html>
- [4] OMNeT++ Homepage. Available: <http://www.omnetpp.org/>
- [5] <http://www.mathworks.com/products/matlab/>
- [6] M.S. Hasan, H. Yu, A. Carrington, and T.C. Yang, "Co-simulation of wireless networked control systems over mobile ad hoc network using SIMULINK and OPNET," IET Communications, vol.3, no. 8, pp. 1297 – 1310, Aug. 2009.
- [7] <http://www.mathworks.com/products/simulink/index-b.html>
- [8] Zhi Zhang¹, Zhonghai Lu¹, Qiang Chen¹, Xiaolang Yan², Li-Rong Zheng¹, „COSMO: CO-Simulation with MATLAB and OMNeT++ for Indoor Wireless Networks“, iPack VINN Excellence Center, Royal Institute of Technology (KTH), Stockholm, Sweden, 2: Institute of VLSI Design, Zhejiang University, Hangzhou, China, IEEE Globecom 2010
- [9] Gilbert E. Perez, Ivica Kostanic, "Comparing a Real-Life WSN Platform Small Network and its OPNET Modeler model using Hypothesis Testing", Government Communications Systems Division, Harris Corporation Melbourne, FL, USA, Electrical and Computer Engineering Department, Florida Institute of Technology, Melbourne, FL, USA, SYSTEMICS, CYBERNETICS AND INFORMATICS VOLUME 12 - NUMBER 7 - YEAR 2014
- [10] Unlocking the Power of OPNET Modeler, Zheng Lu, Hongji Yang, Cambridge University Press, 2012
- [11] MATLAB - The Language of Technical Computing (Application Program Interface Guide v5), The MathWorks Inc, 1998
- [12] MurGee URL: <http://www.murgee.com/>