BLOCKING COMMUNICATION GRID VIA ILL-INTENTIONED TRAFFIC

Zaitsev D.A.[1], Shmeleva T.R.[2]
International Humanitarian University,
Fontanskaya Doroga, 33
Odessa 65009, Ukraine
daze@acm.org[1]
tishtri@rambler.ru[2]

Retschitzegger W.[3], Proll B.[4]
Johannes Kepler University,
Altenbergerstraße 69,
A-4040 Linz, Austria
werner.retschitzegger@jku.at[3]
birgit.proll@jku.at[4]

KEYWORDS
Computing grid, traffic, performance evaluation, colored Petri net, deadlock

ABSTRACT

A rectangular communication grid model was constructed in the form of a colored Petri net as a composition of submodes of terminal devices producing and consuming packets and communication devices switching packets for their delivery among terminal devices. The model was supplied with traffic guns simulating ill-intentioned traffic. Via simulation in CPN Tools it was shown that a traffic duel of a pair of guns brings the grid to a full deadlock with less than ten percent of the peak load. Thus, the vulnerability of grids to traffic attacks was revealed. The future work is aimed to resist the attacks.

1. INTRODUCTION

Grid and cloud computing [1,2] represents an alternative to the centralized supercomputer concept and possesses a series of advantages among which the following should be mentioned: high reliability because a failure or breakdown of a single node influences the operability of the entire grid negligibly; possibility to gather idle computing resources of the entire world; accessibility of information, applications, and processors from any access point.

Wide spreading of grid and cloud computing brings along with considerable advantages some imperfections caused by the vulnerability of rapidly grown structure. So, in [3,4], a possibility of blocking a regular grid structure via ill-intentioned traffic was revealed. Since classical Petri nets were applied, the numerical characteristics such as probability of blocking, percentage of the grid performance fall, and its influence on QoS were not investigated.

Advantages of modeling systems, and, in particular, networks via colored Petri nets (CPN) were discussed in [5,6]. CPN contain facilities for modular (hierarchical) composition of models, description of timed characteristics and statistical processing of simulation results. Colored Petri nets of simulating system CPN Tools were applied successfully to the performance and QoS evaluation of modern networking technology such as Ethernet, MPLS, PBB that is reflected in the kit of real-life examples situated on its site [7].

One of the first works on modeling grids via CPN [8] is devoted to the grid workflows; in [9] special kind of prediction Petri net was introduced; in [10] Petri nets describe coordinated cyber-attacks on grids.

The present work elaborates Petri net grid models [3] bringing them to the class of CPN with the goal of the numerical evaluation of consequences, which ill-intentioned traffic, appearing a threat to the grid operability, affects. As a future research direction, counter measures to resist this threat are planned.

2. COLORED PETRI NET MODEL OF GRID

A colored Petri net, according to [5], represents a Petri net graph whose elements are loaded with a functional programming language CPN ML. Petri net graph consists of places, depicted as ovals, and transitions, depicted as rectangles, connected via arcs. Dynamical elements, named tokens, are situated inside places; they are consumed and produced as a result of transitions' firing. A CPN token is an object of an abstract data type named a color set. Arcs and transitions are loaded with CPN ML predicates and functions representing convenient tools for modeling.

A timed delay could be associated either with a transition or with an arc; delayed tokens wait elapsing time into output places of a transition. To indicate a trace of model behavior, two magnitudes are employed: *Step* – number of fired transitions and *Time* – current value of the model time. Each timed token *k* has a timestamp *t* of its activation written in the form *k@t*; delays are expressed in the form *@+d* that means an increment of the current model time to reach the token activation time. Such a bit complicated scheme allows transitions to fire instantaneously.

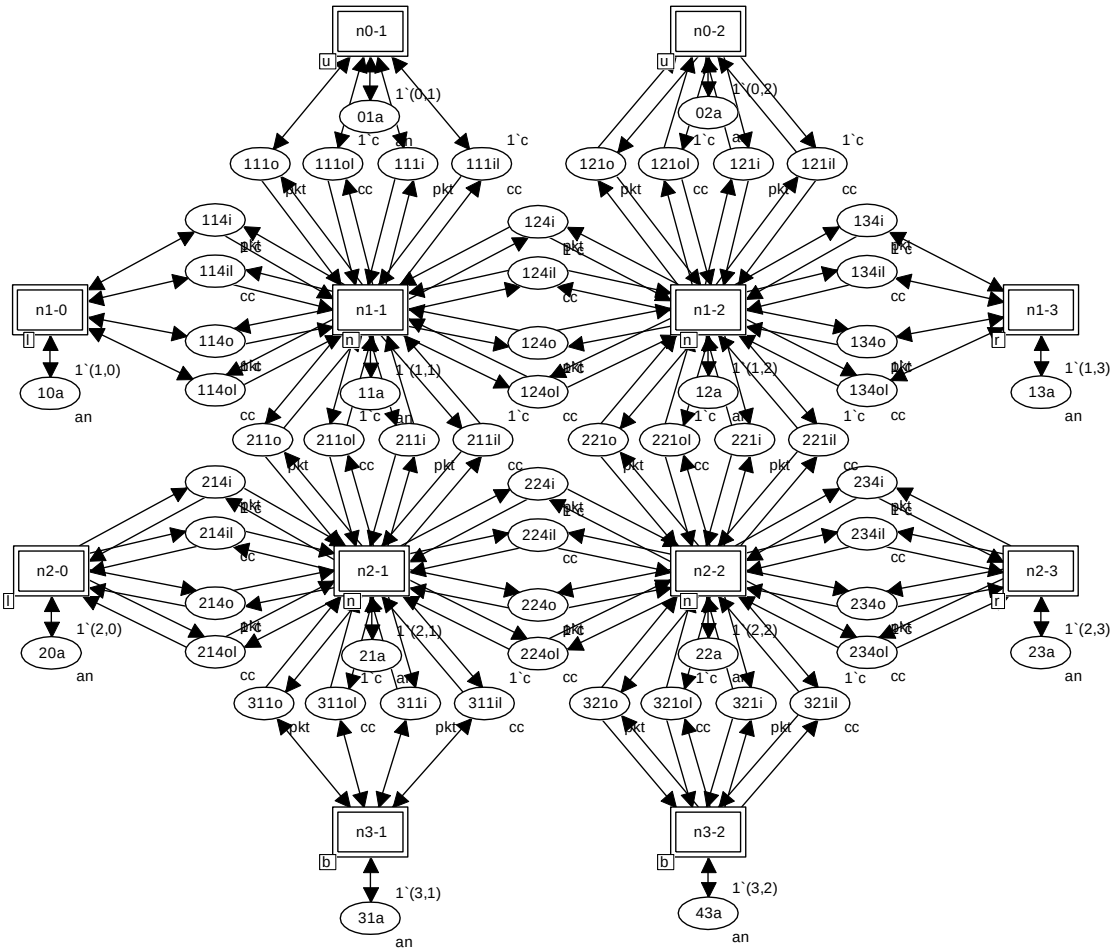
Modular composition of a model is provided via the operation of transition substitution when a transition maps to a subnet. As a subnet interface, contact places, named ports, are indicated with tags *In*, *Out*, and *I/O*. Transition substitution imply attaching a tag of subnet to the transition and mapping ports into sockets – places of higher level net incidental to the transition.

Specific issues of networks' models construction in CPN Tools are studied in [6]; CPN Tools software and example real-life models could be downloaded from [7].

2.1. Composition of Grid Model

Grid model is composed as a square (rectangular) matrix of data communication equipment (DCE) supplied with data terminal equipment (DTE) attached on the borders. Each DCE has four ports situated on the sides of a unit size square and works in full-duplex

mode providing two channels for independent transmitting and receiving packets. DCE implements switching packets among ports based on the store-and-forward principle. DTE produces and consumes packets to model either grid workload or special (ill-intentioned) traffic.



Figures 1: An Example of Square Grid Model with Size 2x2.

Typical DCE model is named *n* (abbreviation of "node"). Typical DTE models are named *l, r, u, b* (abbreviations of "left", "right", "upper", and "bottom" borders' names correspondingly); they have minor difference as it is described in sequel.

We use a system of nodes' addressing within a rectangular grid of size $k1 \times k2$ with two integer numbers *(i,j)* according to Fig. 1 where the first number denotes a row and the second – denotes a column. DCE are enumerated within the range from *1* to *k1* in vertical direction and from 1 to k2 in horizontal direction while DTE have either of indices equal to *0* or *k1+1 (k2+1)*.

Description of the node address has the following form

*colset an = product INT * INT;*

Model of a grid is composed of models of nodes; name of a node contains suffix equaling to its address; each node is supplied with a place with suffix "a" having its address. Type of node is defined by the transition substitution tag written in small rectangle beside it: *n, l, r, u, b*.

A packet is transmitted between a pair of DTE and has the following description

*colset pkt = record da:an * sa:an * co:STRING timed;*

where *da* is a destination address, *sa* is a sender address and a string *co* represents some content of a packet.

To calculate buffers' sizes, we use elementary tokens of the following form

colset cc = unit with c;

Each channel is described by a pair of places: one of type *pkt* to store a packet and another of type *cc* to store the buffer size. Buffer sizes are measured in number of packets; port buffers have size equal to unit. Each port consists of two channels: input with suffix "i" to receive packets and output with suffix "o" to transmit packets. Thus, a port is represented by four mentioned contact places for connection with a neighbor node. Enumeration of ports is done clockwise starting from the upper port whose number is equal to unit.

Connection of nodes to assemble a grid model is implemented via merging corresponding contact places of neighbor devices as shown in Fig 1. In horizontal direction port 2 of the left node is merged with port 4 of the right node; in vertical direction port 3 of the

upper node is merged with port 1 of the bottom node. To avoid duplicity in the port names, we consider only names of the left (number 4) and the upper (number 1) ports of the current node. The right (number 2) and the bottom (number 3) ports' names do not appear within the model; instead of them we use names of the left (number 4) and the upper (number 1) ports of the neighbor nodes. Thus, suffix "i/o" corresponds to the input/output channel of either the left (number 4) or the upper (number 2) port of one of the connected nodes.

2.2. Model of a DCE Node

Model of a DCE node is shown in Fig. 2. It consists of $4 \times 4 = 16$ described above contact places of ports situated on the sides of a square: *p1o, p1ol, p1i, p1il, p2o, p2ol, p2i, p2il, p3i, p3il, p3o, p3ol, p4i, p4il, p4o, p4ol*. The order of places in Fig. 2 provides the connection of channels of opposite types when composing a grid: the input channel is connected with the output channel of a neighbor node and vise versa. Port 2 is connected with port 4 of the right neighbor node and port 3 is connected with port 1 of the bottom neighbor node. Contact place *ma* contains address of the current DCE node.

The internal buffer of a node is represented by the five following places: places *pb1, pb2, pb3, pb4* store packets redirected to the corresponding port while place *pbl* represents the available buffer size. Moreover, places *pb1, pb2, pb3, pb4* are complimentary to place *pbl* that means that storing a packet in either of *pb1, pb2, pb3, pb4* takes a token from *pbl* and extracting packet from either of *pb1, pb2, pb3, pb4* puts a token into *pbl*.

The output channel of a port is modeled by a single transition – for four ports *t1o, t2o, t3o, t4o* correspondingly. For instance, for port 1, transition *t1o* takes a packet from place *pb1* and puts the packet into place *p1o*; besides *t1o* takes a token from *p1ol* and puts a token into *pbl* modeling changes in the buffers' sizes. The condition of transition *t1o* firing includes presence of a packet in the corresponding section of the internal buffer *pb1* and availability of the destination port output buffer – presence of a token in place *p1ol*.

The input channel of a port is modeled by three transitions – a transition for each possible direction of transmission, excluding the packet arrival port; thus the packet redirection decision is modeled. For instance, for port 1, transition *t1i2* models redirection to port 2, *t1i3* – to port 3, *t1i4* – to port 4. Configuration of transition incidental arcs differs by the internal buffer section only. For instance, transition *t1i2* takes a packet from *p1i* and puts the packet into *pb2*; besides *t1i2* takes a token from *pbl* and puts a token into *p1il* modeling changes in the buffers' sizes. The condition of transition *t1i2* firing includes presence of a packet in the port 1 input buffer *p1i* and availability of the internal buffer – presence of a token in place *pbl*.

Switching packets at random is the simplest solution but it does not provide correct packets' delivery according to the destination address. Within the model, the packet switching decision is implemented via four redirection predicates *to1(p,a), to2(p,a), to3(p,a), to4(p,a)*. They are used as guard functions of transitions redirecting a packet to the corresponding port. A pair of predicates can be true for a given packet; in this case the direction choice is done at random among corresponding transitions. The choice could be deterministic, for instance, when a direction with greater difference of addresses is preferred but random choice works better.

The switching algorithm is based on the valid direction of transmission defined by the difference of the current and destination address on horizontal and vertical coordinate axis given by predicates *v*. It's the main rule with exceptions for the borders of the communication grid. When destination DTE and current DCE are on the same border ($db \wedge cb$), the packet should not been delivered to a wrong DTE. The application of the main rule allows transmission in both permitted directions but DTE does not provide packets redirection; thus, the direction to the DTE different from the destination DTE should be invalid. The special case ($db \wedge cb \wedge nb$) represents the direction to the neighbor destination DTE which is permitted.

Redirection predicates use the following auxiliary predicates, where constants *k1* and *k2* define the size of the rectangular grid in the vertical and horizontal directions correspondingly.

Valid direction of transmission:

```
fun v1(p:pkt,a:an)=((#1(#da p))<(#1 a));
fun v3(p:pkt,a:an)=((#1(#da p))>(#1 a));
fun v4(p:pkt,a:an)=((#2(#da p))<(#2 a));
fun v2(p:pkt,a:an)=((#2(#da p))>(#2 a));
```

Belonging of the destination address to the corresponding border:

```
fun db4(p:pkt)=((#2(#da p))=0);
fun db2(p:pkt)=((#2(#da p))=(k2+1));
fun db1(p:pkt)=((#1(#da p))=0);
fun db3(p:pkt)=((#1(#da p))=(k1+1));
```

Belonging of the current DCE to the corresponding border:

```
fun cb4(a:an)=((#2 a)=1);
fun cb2(a:an)=((#2 a)=k2);
fun cb1(a:an)=((#1 a)=1);
fun cb3(a:an)=((#1 a)=k1);
```

Destination of the packet to the corresponding neighbor node:

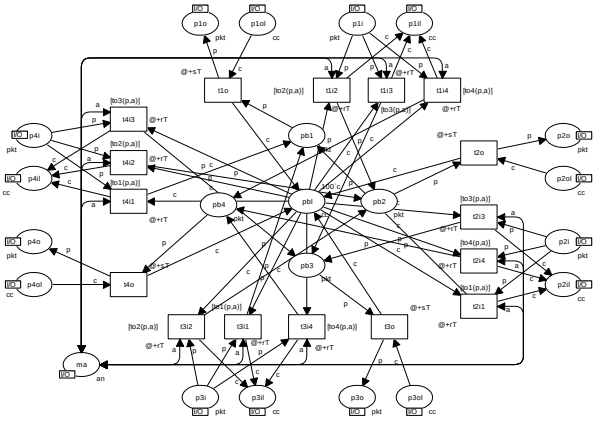
```
fun nb4(p:pkt,a:an)=((#2 a)=(#2 (#da p)+1)) andalso
((#1 a)=(#1 (#da p)));
fun nb2(p:pkt,a:an)=((#2 a)=(#2 (#da p)-1)) andalso
((#1 a)=(#1 (#da p)));
fun nb1(p:pkt,a:an)=((#1 a)=(#1 (#da p)+1)) andalso
((#2 a)=(#2 (#da p)));
fun nb3(p:pkt,a:an)=((#1 a)=(#1 (#da p)-1)) andalso
((#2 a)=(#2 (#da p)));
```

The redirection predicates:

```
fun to1(p:pkt,a:an)=v1(p,a) andalso
((not (db1(p) andalso cb1(a))) orelse (db1(p) andalso
cb1(a) andalso nb1(p,a)));
fun to3(p:pkt,a:an)=v3(p,a) andalso
```

((not (db3(p) andalso cb3(a))) orelse (db3(p) andalso cb3(a) andalso nb3(p,a)));
fun to4(p:pkt,a:an)=v4(p,a) andalso
((not (db4(p) andalso cb4(a))) orelse (db4(p) andalso cb4(a) andalso nb4(p,a)));
fun to2(p:pkt,a:an)=v2(p,a) andalso
((not (db2(p) andalso cb2(a))) orelse (db2(p) andalso cb2(a) andalso nb2(p,a)));

The above predicates represent a result of the model behavior analysis and reflect a certain balance of simplicity with considering local information only and rather good grid performance. They themselves could be studied among other factors influencing grid performance and QoS.



Figures 2: Model of a DCE Node.

Timed characteristics of the model are given by two parameters *sT* and *rT* which represent a timed delay of sending and receiving a packet correspondingly. Remind that according to the time concept of CPN Tools, a transition fires instantaneously but the corresponding timed delay is applied to the output tokens which spend the delay into output places staying in unavailable state.

2.3. Model of a DTE Node

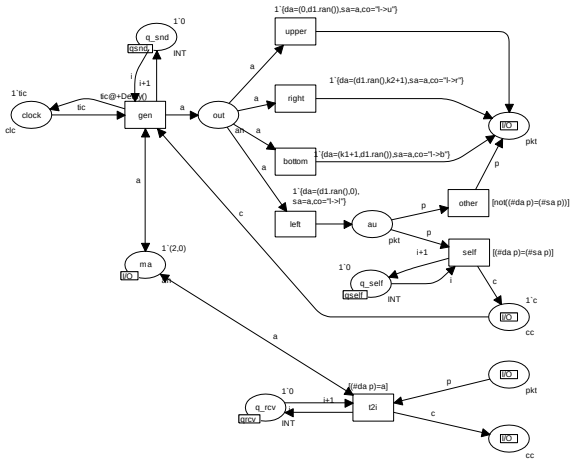
The simplified function of a DTE node is to produce and consume packets; computational aspects of DTE work could be modeled as well but they are left beyond the scope of the present paper. DTE model of a left border node is represented in Fig. 3; it is subdivided into the sending and receiving channels (tracts).

The receiving channel is modeled via transition *t2i* that consumes received packets counting their total number for the entire model in place *q_rcv* of fusion set *qrcv*. For more precise estimations, received and sent packets could be counted separately for each pair of $(2 \cdot k1 + 2 \cdot k2)^2 - (2 \cdot k1 + 2 \cdot k2)$ communicating devices.

The sending channel consists of a timer represented via transition *gen* whose periodicity is controlled by place *clock*; its only token of type *tic* is delayed via random function *Delay()* which defines the period. As a result, a token, equal to the current node address *a*, is put into place *out*; it is a workpiece to produce a packet.

Either of alternative transitions *left, upper, right, bottom*, which are chosen on random, fires generating a packet directed to the corresponding border; its destination address is chosen on random within the border address range via standard function *ran()*; auxiliary types *d1* and *d2* have the following description

colset d1=int with 1..k1;
colset d2=int with 1..k2;



Figures 3: Model of a DTE Node (left border).

The only difference of four types of the border nodes *l, u, r, b* consists in filtering its own destination address which is done after the corresponding transition *left, upper, right, bottom*. In Fig. 3, a model of a left border node is represented, so the filtering is done after transition *left* that puts generated packet into intermediate place *au*. Transition *self* extracts and consumes packets directed to the current device counting their number for the entire model in place *q_self* of fusion set *qself* while transition *other* puts other packets to the output channel buffer of the device.

For random choice of alternative transitions and for implementation of standard function *ran(),* CPN Tools uses the uniform distribution. Besides, CPN Tools offers a wide range of known distributions to describe user random functions. For, instance, a Poisson distribution is chosen for function *Delay()* and the corresponding description has the following form

fun Delay()=poisson(10.0);

3. SIMULATING GRID WORKLOAD

Simulating the grid workload, produced by DTE models *l, u, r, b*, allows debugging the entire model, estimating the influence of its parameters on the grid performance, and studying the grid behavior under the peak load.

Basic parameters of the model, whose influence on the grid behavior was studied, are:

- DCE internal buffer size *bs*;
- intensity of the workload – a parameter of DTE model Poisson distribution *wl*;
- performance of DCE – timed delays of *rT* and *sT* of packet receiving and sending.

At small values of the internal buffer size *bs*, for instance, equal to 10, the grid falls into a deadlock even at traffic equal to 10% of the grid bandwidth. Rather bulky buffer, for instance, of size 10K and more does not allow observing a deadlock even at the peak load. In majority of simulations, the buffer size equal to 100 was chosen that allows observing either deadlocks or their absence on feasible intervals of time.

Table 1 shows that the grid is brought to a deadlock even via regular workload when the buffer size equals to 100. In majority of cases, a deadlock means that buffers of some DCE are full and the difference of the numbers of the sent and received packets is approaching the overall grid buffer size *k1*k2*bs* that equals to 6400 for Table 1 working parameters.

Table 1: Bringing the Grid to a Full Deadlock via Workload.

Workload intensity (*wl*)	*Step*	*Time*	Number of sent packets *qsnd-qself*	Number of received packets *qrcv*	Grid performance *gp* (packets/MTU)
50.0	10000000	854654	529965	529918	0,62
30.0	10000000	512757	529899	529825	1,03
20.0	10000000	341850	530026	529900	1,55
10.0	10000000	170992	530350	547421	3,1
9.0	120000000	1845409	6357110	6356608	3,44
8.0	607161*	12368	34817	29897	2,42
7.0	336234*	6413	20163	15584	2,43
6.0	200348*	2963	12659	9123	3,07
5.0	142266*	1876	8499	6261	3,34
4.0	99624*	1233	7219	3925	3,18
3.0	78952*	877	6233	2621	2,99
2.0	64852*	765	5668	1596	2,09

rT=sT=5, bs=100, k1=8, k2=8;

** – the grid comes to a full deadlock – no permitted transitions.*

Thus, the peak load of the grid at *rT=sT=5* is about *gp=3* packets/MTU that is reached when the workload intensity is about *wl=10.0*. The abbreviation MTU means "model time unit" whose value could be chosen when scaling timed characteristics of a real-life network [6]. Note that, at a boundary workload *wl=9.0*, more prolonged simulation was done but a deadlock was not reached. When approaching a full deadlock, the grid performance can decrease in spite of high intensity of traffic because of partial intermediate deadlocks slowing down the packets delivery. For further investigation of the grid behavior, the workload of 30% is chosen that is achieved at *wl=30.0*.

4. SIMULATING ILL-INTENTIONED TRAFFIC

To simulate ill-intentioned traffic, a model of packets' gun is constructed and its copies are connected to the grid borders. The parameters, whose influence on the model behavior is estimated, are the number and location of guns, their targets, and intensity of guns' work.

A simple model of a packets' gun *g* is shown in Fig. 4. Its work resembles the sending channel of DTE (Fig. 3) but both, source *a* and destination (target) *ta* addresses are given by the marking of external places *ga* and *ta* correspondingly. The periodicity of its shots is given by the random function *gDelay()* which distribution is Poisson also. The number of generated packets (shots fired) is counted in place *q_sndg* of fusion set *qsndg*. Note that, consuming of tokens, generated by a gun, is provided by the regular DTE models.

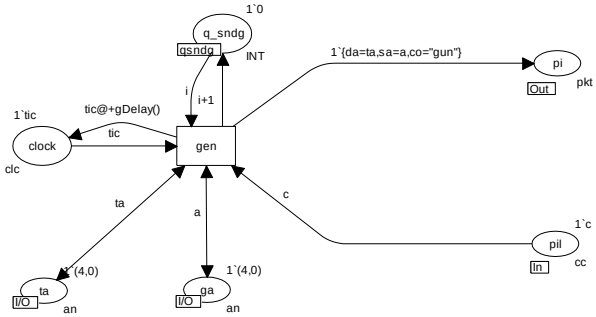
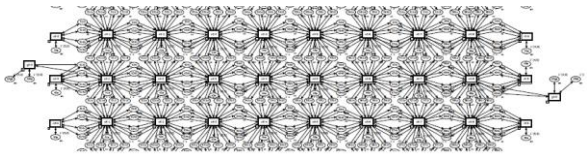


Figure 4: Model of Packets' Gun.

The regular grid model was supplied with a few packet guns, and their influence on the model behavior was studied. As the main characteristic, the time interval for bringing the model to a full deadlock was considered. Attaching a gun or a few guns with different targets influenced the model behavior insignificantly. The most significant result was achieved via attaching a pair of guns with mutual targets – a traffic duel. An example of a traffic duel is shown on a fragment of the grid model in Fig. 5; it was denoted as *(4,0)<->(4,9)* indicating the nodes of guns' attachment and their targets. The detailed characteristics of simulation are put in Table 2. Thus, a traffic duel causes a deadlock with an extra load of about 10%.



Figures 5: A Traffic Duel *(4,0)<->(4,9)* on a Grid Fragment.

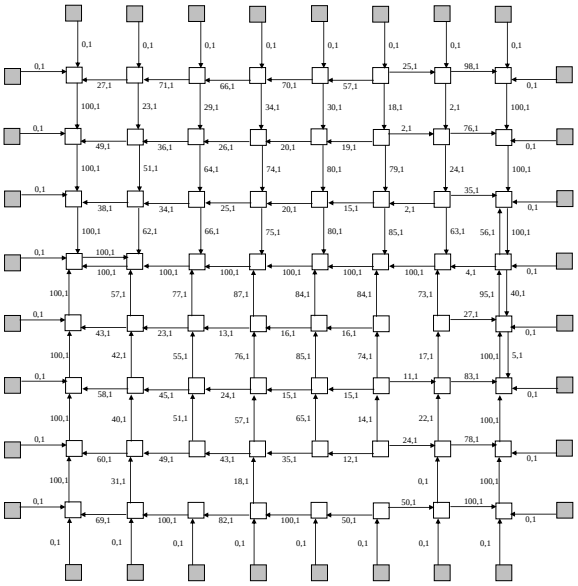
Table 2: Bringing the Grid to a Full Deadlock via the Traffic Duel.

Traffic guns' intensity (*wl*)	*Step*	*Time*	Number of sent packets *qsnd-qself*	Number of received packets *qrcv*	Grid performance *gp* (packets/MTU)
10.0	20000000	865475	1067732	1067631	1,24
6.0	20000000	784017	1071955	1071786	1,36
5.0	255240*	26556	18153	11609	0,43
4.0	188828*	23432	14515	7971	0,34

rT=sT=5, bs=100, wl=30.0, k1=8, k2=8;

** – the grid comes to a full deadlock – no permitted transitions.*

Various other locations of guns did not produce deadlocks faster than the above studied median except of its vertical orientation. For instance, a diagonal traffic duel (1,0)<->(8,9) requires on average 8% more time to fall in a full deadlock. An example of a full deadlock is shown in Fig. 6, where inscriptions on the arcs indicate the number of packets in the corresponding section of the internal buffer and, after comma, the number of packets in the port buffer.



Figures 6: An Example of a Full Deadlock *(bs=100, wl=10.0, gl=5.0, td=(4,0)<->(4,9), Step=290529, Time=10941, k1=8, k2=8).*

Thus, simulation results acknowledged the hypothesis advanced in [3,4] that a grid could be blocked via ill-intentioned traffic. In the simplest case, two traffic generators are required ensuing a traffic duel with load about 10% of the grid peak load.

5. CONCLUSIONS

Issues of the grid blocking via ill-intentioned traffic were investigated using simulation in the environment of modeling system CPN Tools. It was shown that even at rather low workload of the grid about 30%, a grid can be entirely blocked with a full deadlock of DCE via a traffic duel with additional load little than 10%. Thus, the vulnerability of the grid structures to the attacks via traffic was revealed.

A future research direction is bringing more realism to the model and investigation of modern architectural solutions for DCE such as cut-through principle of packets' switching. Real-life devices overcome deadlocks, either local or global, using two rather simple features: they drop packets entering a busy device and clean flooded buffers tracking packets' ageing time. Thus the described deadlocks last for a short interval of time representing a cause of the performance fall.

The ultimate goal of the future work is counter measures to detect and resist traffic attacks on grids avoiding significant performance fall.

REFERENCES

1. *Grid and Cloud Computing: Concepts, Methodologies, Tools and Applications* (4 Vol.), Information Resources Management Association (USA), IGI-Global, 2012, 2134 pp.
2. Preve, N.P. (Ed.). *Grid Computing: Towards a Global Interconnected Infrastructure*. Springer, 2011, 312 p.
3. Shmeleva T.R., Zaitsev D.A., Zaitsev I.D. Analysis of Square Communication Grids via Infinite Petri Nets. *Transactions of Odessa National Academy of Telecommunication*, no. 1 (2009) 27-35. http://archive.nbuv.gov.ua/portal/natural/Nponaz/2009_1/files/20091_SmelevaZaycevZaycev.pdf
4. Zaitsev D.A., Shmeleva T.R. Verification of hypercube communication structures via parametric Petri nets. *Cybernetics and Systems Analysis*, Volume 46, Number 1 (2010), 105-114, doi: 10.1007/s10559-010-9189-y
5. Jensen, K., Kristensen, L.M. *Coloured Petri Nets: Modelling and Validation of Concurrent Systems*. Springer, 2009, 384p.
6. Zaitsev, D.A. *Clans of Petri Nets: Verification of protocols and performance evaluation of networks*. LAP LAMBERT Academic Publishing, 2013, 292 p.
7. http://cpntools.org
8. Bratosin, C.C., Aalst, W.M.P. van der, Sidorova, N.. Modeling grid workflows with colored Petri nets. In K. Jensen (Ed.), *Proceedings of the 8th Workshop and Tutorial on Practical Use of Coloured Petri Nets and the CPN Tools*, 22-24 October 2007, Aarhus, Denmark, pp. 67-86.
9. Retschitzegger W., Baumgartner N., Gottesheim W., Mitsch S., Schwinger W. Situation Prediction Nets – Playing the Token Game for Ontology-Driven Situation Awareness. *Proceedings of 29th International Conference on Conceptual Modeling (ER)*, Vancouver, Nov. 2010.
10. Chen, T.M., Sanchez-Aarnoutse, J.C., Buford, J. Petri Net Modeling of Cyber-Physical Attacks on Smart Grid. *IEEE Transactions on Smart Grid*, Vol. 2 , No. 4, 2011, 741 – 749.