A COMPARISON OF EFFICIENT ALGORITHMS FOR SCHEDULING PARALLEL DATA REDISTRIBUTION

Marios-Evangelos Kogias and Timotheos Aslanidis

National Technical University of Athens, Athens, Greece

ABSTRACT

Data redistribution in parallel is an often-addressed issue in modern computer networks. In this context, we study the case of data redistribution over a switching network. Data from the source stations need to be transferred to the destination stations in the minimum time possible. Unfortunately the time required to complete the transfer is burdened by each switching and thus producing an optimal schedule is proven to be computationally intractable. For the purposes of this paper we consider two algorithms, which have been proved to be very efficient in the past. To get improved results in comparison to previous approaches, we propose splitting the data in two clusters depending on the size of the data to be transferred. To prove the efficiency of our approach we ran experiments on all three algorithms, comparing the time span of the schedules produced as well as the running times to produce those schedules. The test cases we ran indicate that not only our newly proposed algorithm yields better results in terms of the schedule produced but runs faster as well.

KEYWORDS

parallel, distributed, package, scheduling, setup cost, approximation, preemption

1. INTRODUCTION

As the need for communication and dissemination of information increases in modern technology based societies, so does the need for faster and more efficient networks and routing of packages between stations. In this context the study of parallel machines (parallel supercomputers, symmetric multiprocessors, multicomputer clusters and IP router switch fabrics), communication backbones interconnecting the machines and the transfer of large packets of data between them, has become an issue of major importance. As information loads continue to increase rapidly, it is expected that the need for well scheduled data transfers that decrease time and resource usage, will keep on being an often addressed subject for many computer scientists and engineers.

In this manuscript and in the context of redistributing data between parallel machines using a centralized switch (Figure1), the problem studied is the Preemptive Bipartite Scheduling problem (encountered as PBS in the literature). Given a set of n_1 source stations and a set of n_2 destination stations, we are required to send across data streams, each initiated from a specific source, to reach an also prespecified target. The duration of each data stream is also predetermined for all data streams scheduled for transfer. Restrictions in the systems considered, are that no source may send data towards more than one destination at any time, nor may a destination station receive, from more than one source station at any time. Messages are sent in packages and to enhance transfer speed we are allowed to preempt any package and continue transfer of any part of that package at a later time. Unfortunately, since the system has to reconfigure after each preemption,

DOI : 10.5121/ijcnc.2014.6304

any interruption of packages transfer will come with a time cost. Information on the data remains that were not transferred along with the main part of the package, have to be saved and a new setup has to be initiated for the next package to start transferring. Consequently prior to sending any of the packages there will be a setup overhead. We consider this overhead to be constant for all transfer initiations. In this paper we aim to minimize the duration of the aforementioned process.

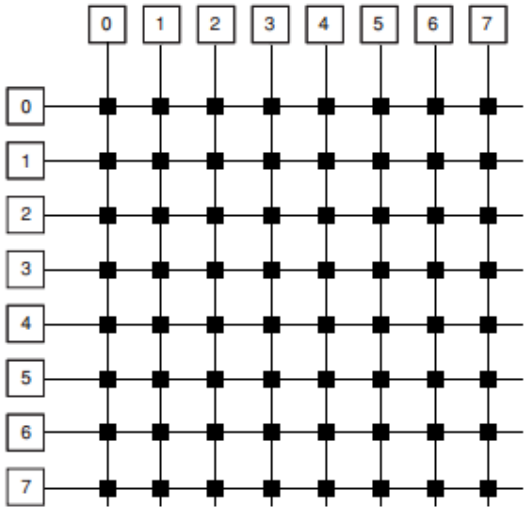

Figure 1. Crossbar switch interconnection network

2. PREVIOUS RESULTS

PBS has been proven to be NP-Hard in [8], and 4/3-ε inapproximable for any ε>0, unless P=NP in [4].

As PBS algorithms can be implemented in various applications, many polynomial time algorithms have been designed to produce solutions close to the optimal, found in [1], [4], [5], [6]. The optimal approximation ratio proven so far is $2-\frac{1}{d+1}$, where d is the setup cost. Proof of that can be found in [1]. Tests on the performance of numerous algorithms producing near-optimal solutions are presented in [4], [5], [6] and [10].

The optimal solution can be found in polynomial time if the setup cost is zero and in the case we are only interested in minimizing the number of preemptions [8]. Finally, in [1] can be found a polynomial time algorithm to calculate the optimal solution for a class of graphs that the authors define.

The performance of our algorithm is compared to that of 2 algorithms found in bibliography:

- A-PBS(d+1), which has the best approximation ratio proven so far [1].
- A1, which appears to perform very well in practice [6].

To schedule each data package, A-PBS(d+1) rounds up the time of each transfer to the closest multiple of d+1 and calculates the package reducing the workload of each station to the minimum multiple of d+1.

Whereas, A1 computes an arbitrary package with a maximum number of data transfers and decides how to preempt by calculating a lower bound to the transfer duration of the remaining packages plus the cost of the current package, to be the minimum possible.

3. GRAPH REPRESENTATION AND NOTATIONS

Our data representation will be through a bipartite graph G(V,U,E). V will be the set of source stations, U the set of receiving stations while E, the set of edges, will correspond to the data loads that have to be transferred from V to U. A weight (or cost) c(v,u), will be assigned to each of the edges e=(v,u), to denote the time required to transfer the data from node v to node u. Edge weights are considered to be non-negative integers.

Furthermore the following notation will be used: $\Delta=\Delta(G)=\max\{\max_{v\in V}(\deg(v)), \max_{u\in U}(\deg(u))\}$, that is, Δ will denote the maximum number of data transfers from or to any of the stations.
The function t: $V\cup U\rightarrow Z_+$ will denote the total workload of any station, namely $t(v)=\sum_{u\in U} c(v,u)$ for any $v\in V$ or $t(u)=\sum_{v\in V} c(v,u)$ for any $u\in U$.
$W=W(G)=\max\{\max_{v\in V}(t(v)), \max_{u\in U}(t(u))\}$, that is W will denote the maximum total transfer time from or to any station.
$d\in Z^*_+$ will denote the overhead to start the next transfer.
The objective function to be minimized is $C(G,d)=\sum_{i=1}^{S} T(p_i)+d\cdot S$, where S is the number of data packages and $T(p_i)$ is the time required for each package p_i, to be transferred.

A lower bound to the optimal solution is $W(G)+d\cdot\Delta(G)$. Yet, this lower bound is not always achievable as shown in [6].

4. THE SPLIT GRAPH ALGORITHM

For the purposes of our algorithm the initial graph is split in two parts. G_M comprises edges of weight at least d and G_m contains all edges of weight less than d. Our main concern for G_M is to keep reducing the workload for each of the stations, achieving the minimum transfer time possible, whereas in the case of G_m, where edge weights are small in comparison to d, we aim in minimizing the number of preemptions. The intuition in designing this algorithm is that for data transfers of long duration, priority on how to schedule has the duration of the data transfer rather than the number of preemptions, whilst for data transfers of shortest duration prioritized is the minimization of the number of preemptions. In particular:

The Split-Graph Algorithm (SGA)

Step 1: Split the initial graph G(V,U,E) in two bipartite graphs $G_m(V_m,U_m,E_m)$ and $G_M(V_M,U_M,E_M)$, where $V_m=V_M=V$, $U_m=U_M=U$ and E_m contains all edges of weight less than d, E_M contains all edges of weight d or more. Clearly in this initiation step $E=E_m\cup E_M$ and $E_m\cap E_M=\varnothing$.
Step 2: Use subroutine1 to find a maximal matching M, in G_M.
Step 3: Use subroutine 2 to calculate the weight of the matching to be removed. Remove the corresponding parts of the edges.
Step 4: If possible, add edges to M, from E_m to maximize |M| and remove them from E_m.
Step 5: Move edges of weight less than d, from the graph induced by step 3 to E_m.
Step 6: Repeat steps 2 to 5 until all edges initially in E_M have been completely removed.
Step 7: Use subroutine 3 to calculate Δ_m maximum matchings in G_m, where Δ_m is the degree of G_m.

Step 8: Schedule the data packets as calculated in steps 2, 3 and 7.

Subroutine 1:

Step 1: M=∅ (Initialization of the matching).
Step 2: For each node w∈ $V_M \cup U_M$ calculate t(w).
Step 3: Sort all nodes w∈ $V_M \cup U_M$ in decreasing order of t(w). Let L be the induced list of nodes.
Step 4: Let w_0 be the 1st node to appear in L. Run sequential search in L to find the 1st neighbour of w_0 appearing in L. Denote that neighbour by w_1.
Step 5: M←M∪{w_0, w_1}.
Step 6: Remove w_0, w_1 from L.
Step 7: Repeat steps 2, to 6 until M becomes maximal.

Subroutine 2:

Step 1: For each edge e=(v,u) of the matching M, with corresponding weight c(e) calculate what the value $W(G_e)$ of the induced graph $G_e(V_e,U_e,E_e)$ would be if all edge weights in the matching were to be reduced by c(e). In this case edges in M, of cost less than c(e) would be completely removed.
Step2: For all e∈M, set r(e)=c(e)+$W(G_e)$ and denote by e_0 the edge such that $r(e_0)=\min\{r(e)|e\in M\}$ and $c(e_0)=\max\{c(e)|r(e)=r(e_0)\}$.
Step 3: For each edge e, in M set its new weight $c(e)=\begin{cases} c(e)-c(e_0), & \text{if } c(e)>c(e_0) \\ 0, & \text{otherwise} \end{cases}$.

Subroutine 3:

Step 1: Add nodes and edges to make G_m a regular graph of degree Δ_m. New edges will be of zero weight. In a regular graph, all nodes will be of the same degree.
Step 2: Calculate a maximum matching M_m in G_m and remove all edges of M_m from G_m. G_m's degree will now be reduced by 1.
Step 3: Repeat step 2 until G_m=∅.
Example:
Consider matrix A to be the adjacency matrix of G(V, U, E), where V={a_1, b_1, c_1, d_1}, U={a_2, b_2, c_2}. Rows 1, 2, 3 and 4, show the times required to transfer the data from nodes a_1, b_1, c_1, d_1 respectively, while columns 1, 2 and 3 stand for the durations needed to transfer the data to a_2, b_2, c_2, respectively. Suppose d=3.

$$A=\begin{pmatrix} 5 & 3 & 2 \\ 1 & 4 & 3 \\ 2 & 5 & 0 \\ 4 & 2 & 5 \end{pmatrix}$$

Let A_M be the adjacency matrix corresponding to G_M and A_m be the adjacency matrix of G_m. Then:

$$A_M=\begin{pmatrix} 5 & 3 & 0 \\ 0 & 4 & 3 \\ 0 & 5 & 0 \\ 4 & 0 & 5 \end{pmatrix} \quad A_m=\begin{pmatrix} 0 & 0 & 2 \\ 1 & 0 & 0 \\ 2 & 0 & 0 \\ 0 & 2 & 0 \end{pmatrix}$$

$t(a_1)=8$, $t(b_1)=7$, $t(c_1)=5$, $t(d_1)=9$, $t(a_2)=9$, $t(b_2)=12$, $t(c_2)=8$.
sorting the nodes: b_2, d_1, a_2, a_1, c_2, b_1, c_1.
$M=\{\{b_2,a_1\}, \{d_1,c_2\}\}$, $e_0=\{d_1,c_2\}$, $r(e_0)=14$.
Add edge $\{b_1,a_2\} \in E_m$ to M. Schedule a transfer of duration 5
Total time for this iteration including preemption cost is 8
The induced matrices after the 1st iteration are:

$$A_M = \begin{pmatrix} 5 & 0 & 0 \\ 0 & 4 & 3 \\ 0 & 5 & 0 \\ 4 & 0 & 0 \end{pmatrix}, \; A_m = \begin{pmatrix} 0 & 0 & 2 \\ 0 & 0 & 0 \\ 2 & 0 & 0 \\ 0 & 2 & 0 \end{pmatrix}$$

$t(a_1)=5$, $t(b_1)=7$, $t(c_1)=5$, $t(d_1)=4$, $t(a_2)=9$, $t(b_2)=9$, $t(c_2)=3$
sorting the nodes: a_2, b_2, b_1, a_1, c_1, d_1, c_2
$M=\{\{a_2,a_1\},\{b_2,b_1\}\}$ $e_0=\{a_2,a_1\}$, $r(e_0)=10$.
No additional edge from E_m. Schedule a transfer of duration 5
Total time for this iteration including preemption cost is 8
The induced matrices after the 2nd iteration are:

$$A_M = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 3 \\ 0 & 5 & 0 \\ 4 & 0 & 0 \end{pmatrix}, \; A_m = \begin{pmatrix} 0 & 0 & 2 \\ 0 & 0 & 0 \\ 2 & 0 & 0 \\ 0 & 2 & 0 \end{pmatrix}$$

$t(a_1)=0$, $t(b_1)=3$, $t(c_1)=5$, $t(d_1)=4$, $t(a_2)=4$, $t(b_2)=5$, $t(c_2)=3$
sorting the nodes: c_1, b_2, d_1, a_2, b_1, c_2, a_1.
$M=\{\{c_1,b_2\},\{d_1,a_2\},\{b_1,c_2\}\}$ $e_0=\{c_1,b_2\}$, $r(e_0)=5$.
No additional edge from E_m. Schedule a transfer of duration 5
Total time for this iteration including preemption cost is 8.
All edges in E_M are now removed, proceeding to step 7 of SGA:

$$A_m = \begin{pmatrix} 0 & 0 & 2 \\ 0 & 0 & 0 \\ 2 & 0 & 0 \\ 0 & 2 & 0 \end{pmatrix}$$

The corresponding maximum matching is $M=\{\{a_1,c_2\}, \{c_1,a_2\}, \{d_1,b_2\}\}$
The duration of the scheduled transfer is 2.
Total time for this iteration including preemption cost is 5.

Total time to transfer the data is 29, while the lower bound is 26.

5. COMPARING THE SCHEDULES

One thousand test cases have been ran for a 15 source-15 destination system for values of setup cost varying from 1 to 100 and message durations varying from 0 to 50. Figure 2 represents each algorithm's performance in terms of proximity to the lower bound. SGA performs significantly

better than both A1 and A-PBS(d+1) and as the overhead increases it shows an increasingly improved performance. It is important to mention that in practice, as information loads exponentially increase, the number of stations and communication tasks increases and so does the setup cost. That is in fact the most encountered situation nowadays.

Figure 3 presents the worst performance that each algorithm had depending on the setup cost, in terms of proximity to the lower bound again. SGA is found to be once again a lot more efficient. Furthermore, SGA in most cases has a worst case really close to the average showing that its performance does not fluctuate much, making it an all cases reliable tool for this type of scheduling.

It is important to stress out that the ratio calculated, is in comparison to the lower bound and not to the optimal value of the objective function. Therefore the schedules produced are in fact a lot closer to the optimal and the ratios a lot closer to 1.

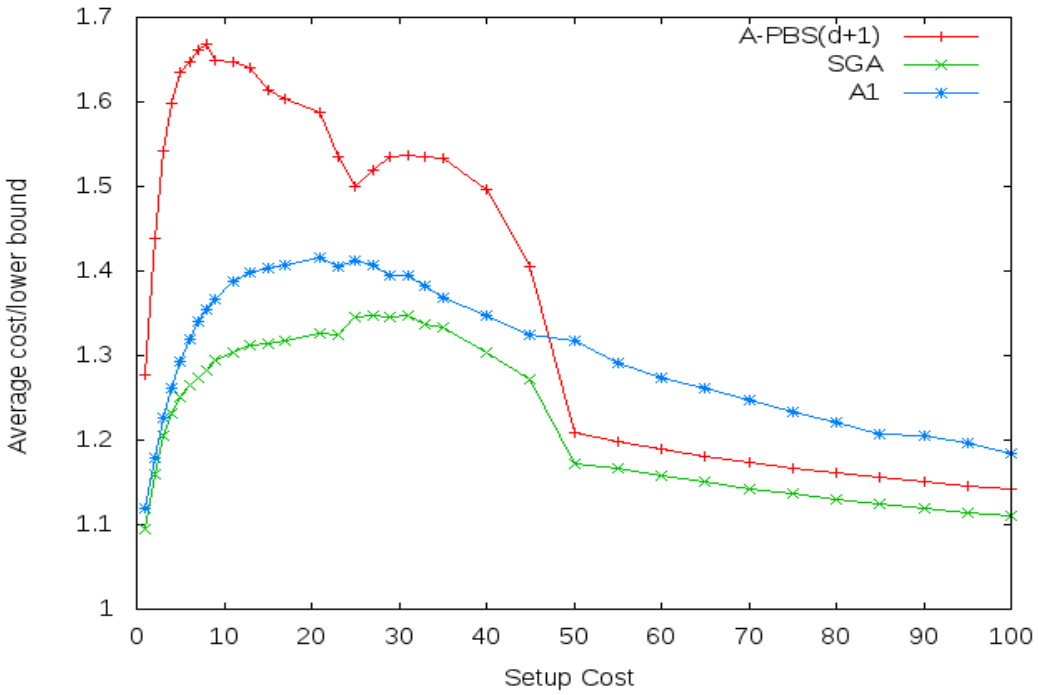

Figure 2. Average (cost/lower bound) comparison

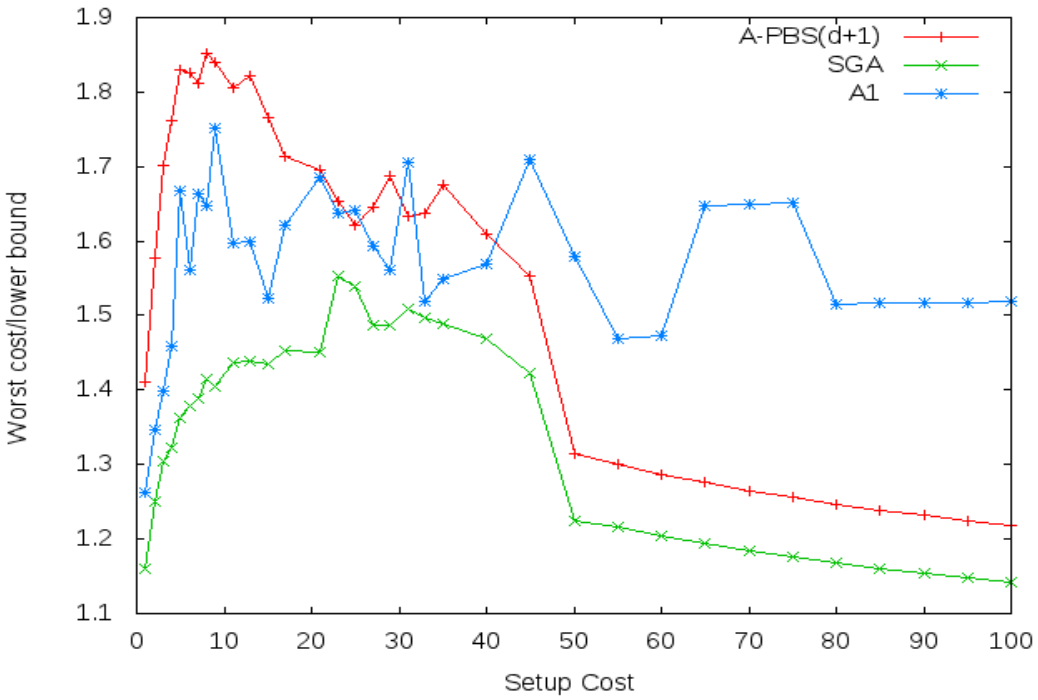

Figure 3. Worst (cost/lower bound) comparison

6. COMPARING THE RUNNING TIMES

To compare time complexities for the 3 algorithms we will assume that |E|=m and we will denote max{|V|,|U|}=n.

All three algorithms compared, will need O(m) iterations to produce a schedule as it is certain that each iteration will result in the complete removal of at least one edge.

For each iteration the most time consuming subroutine of A-PBS(d+1) is finding a maximum matching which will need $O(\sqrt{n}\, m)$ [12]. Therefore the overall complexity of A-PBS(d+1) is $O(\sqrt{n}\, m^2)$

As far as A_1 is concerned, determining how to preempt is the most time consuming subroutine for each iteration. For each of the edge weights in the matching, it calculates for each node v, deg(v) and t(v). Each of these three processes requires O(n), therefore the complexity of each iteration is $O(n^3)$ making the overall time complexity $O(n^3m)$.

Finally, the complexity of SGA is determined by the maximum matching process that takes place in step2 of subroutine3 and consequently the overall complexity is the same as that of A-PBS(d+1), namely $O(\sqrt{n}\, m^2)$.

As |E| varies from O(n) to $O(n^2)$, it is clear that A-PBS(d+1) and SGA work faster than A_1. Yet, SGA will only be using the maximum matching algorithm after most of the graphs edges are removed. Consequently in practice it runs somewhat faster.

7. CONCLUSIONS AND FUTURE WORK

Our newly presented algorithm (SGA), has proven to produce more efficient schedules. We believe that the idea of splitting the initial graph in parts can be further researched and depending on the magnitude of the edges as well as the setup cost, the Split-Graph Algorithm's efficiency can be further improved. An approximation ratio for SGA could be established to be less than 2. Exploiting to a greater extend algorithms that provide optimal solutions for special instances of the problem might also yield interesting new approximation algorithms. Finally, lifting limitations of the problem or introducing new ones could help in identifying new classes of graphs for which polynomial algorithms might provide an optimal schedule.

REFERENCES

[1] F. Afrati, T. Aslanidis, E. Bampis, I. Milis, Scheduling in Switching Networks with Set-up Delays. Journal of Combinatorial Optimization, vol. 9, issue 1, p.49-57, Feb 2005.

[2] P. B. Bhat, V. K. Prasanna and C. S. Raghavendra, Block Cyclic Redistribution over Heterogeneous Networks, in 11th International Conference on Parallel and Distributed Computing Systems (PDCS 1998), 1998.

[3] G. Bongiovanni, D. Coppersmith and C. K.Wong, An optimal time slot assignment for an SS/TDMA system with variable number of transponders, IEEE Trans. Commun. vol. 29, p. 721-726, 1981.

[4] J. Cohen, E. Jeannot, N. Padoy and F. Wagner, Messages Scheduling for Parallel Data Redistribution between Clusters, IEEE Transactions on Parallel and Distributed Systems, vol. 17, Number 10, p. 1163, 2006.

[5] J. Cohen, E. Jeannot, N. Padoy, Parallel Data Redistribution Over a Backbone, Technical Report RR-4725, INRIA-Lorraine, February 2003.

[6] P. Crescenzi, X. Deng, C. H. Papadimitriou, On approximating a scheduling problem, Journal of Combinatorial Optimization, vol. 5, p. 287-297, 2001.

[7] I. S. Gopal, G. Bongiovanni, M. A. Bonucelli, D. T. Tang, C. K. Wong, An optimal switching algorithm for multibeam satellite systems with variable bandwidth beams, IEEE Trans. Commun. vol. 30, p. 2475-2481, Nov. 1982.

[8] I. S. Gopal, C. K. Wong, Minimizing the number of switchings in an SS/TDMA system IEEE Trans. Commun. vol. 33, p. 497-501, 1985.

[9] T. Inukai, An efficient SS/TDMA time slot assignment algorithm IEEE Trans. Commun. vol 27, p. 1449-1455, Oct. 1979.

[10] E. Jeannot and F. Wagner, Two fast and efficient message scheduling algorithms for data redistribution over a backbone, 18th International Parallel and Distributed Processing Symposium, 2004.

[11] A. Kesselman and K. Kogan, Nonpreemptive Scheduling of Optical Switches, IEEE Transactions in Communications, vol. 55, number 6, p. 1212, 2007.

[12] S. Micali and V. V. Vazirani, An $O(\sqrt{|V|}|E|)$ algorithm for finding a maximum matching in general graphs, in Proc. 21st Ann IEEE Symp. Foundations of Computer Science, p. 17-27, 1980.

[13] K. S. Natarajan and S. B. Calo, Time slot assignment in an SS/TDMA system with minimum switchings IBM Res. Rep. 1981.

[14] B. Towles and W. J. Dally, Guaranteed Scheduling of Switches with Configuration Overhead, in Proc. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies INFOCOM '02. pp. 342-351, June 2002.

Authors

Timotheos Aslanidis was born in Athens, Greece in 1974. He received his Mathematics degree from the University of Athens in 1997 and a master's degree in computer science in 2001. He is currently a phd candidate at the National and Technical University of Athens in the department of electrical and computer engineering. His research interests comprise but are not limited to computer theory, number theory, network algorithms and data mining algorithms.

Marios-Evangelos Kogias was born in Athens in 1991. Currently, he is an undergraduate student at the National Technical University of Athens pursuing a diploma in Electrical and Computer Engineering. He is a passionate coder and interested mainly in operating systems, computer networks and algorithms.