PARALLEL ALGORITHMS FOR MULTICORE ROUTERS IN LARGE COMPUTER NETWORKS - A REVIEW

Andrzej Karbowski [*,**]

[*] *NASK (Research and Academic Computer Network), ul. Wąwozowa 18, 02-796 Warsaw, Poland*
[**] *Institute of Control and Computation Engineering, Warsaw University of Technology, ul. Nowowiejska 15/19, 00-665 Warsaw, Poland,*
E-mail: A.Karbowski@ia.pw.edu.pl

Abstract: In the paper several classical and new algorithms for calculation of routing tables in large networks, e.g.: Dijkstra, Bellman-Ford, SLF-LLL, auction, Q-routing and ant(swarm)-routing are compared - both theoretically and computationally. The special attention is paid to the possibility of making use of current multicore processors in routers and the effectiveness of the parallel implementations. *Copyright © 2007 IFAC*

Keywords: Routing-algorithms, Computer Networks, Optimization problems, Adaptive algorithms

1. INTRODUCTION

In the last 3-4 years a noticeable change in computer hardware took place. Namely, because of the heat problems, the clock frequency of processors stopped to double every 1.5 year and instead, in a single processor die we have several logically independent CPUs, so-called cores. Since the number of computers connected to networks is continually growing, so the number of computers in big domains, to keep up the speed of the routing algorithms, it is necessary to apply parallel calculations of the routing tables. Otherwise, they will adapt to the changing situation in the networks slower and slower, and the Quality of Service of the networks will decrease.

Fortunately, computers with current dual or quad core processors are shared memory machines, what means, that to write the parallel programs a programmer should only know software tools for concurrent computing, that is for thread programming [1], such as: POSIX threads or OpenMP directives.

One may distinguish two classes of dynamic routing algorithms in computer networks (Perlman, 1999): distance-vector and link-state algorithms. The former are actually fine-grain type and consist in exchanging between the neighboring routers during the calculations the information concerning the estimates of the costs of the best connections to remote routers. At the beginning the routers had no knowledge about the topology and the state of the network (including the current costs of transmission through arcs), they only know their neighbors and the costs of sending packets to them. A representative of this approach is the Bellman-Ford algorithm implemented in the RIP protocol. Because during the optimization all routers have to maintain the tables with current

[1] Using `fork` function from UNIX system library and IPC library is also possible, but not so convenient.

estimates of - usually - the shortest distances to all routers and often communicate with all neighbors, this approach scales badly and is efficient only in small networks.

In the link-state protocols every router has the same global data concerning the state of the whole network, that is: the complete graph, all current costs of transmission along arcs, and calculates independently the best routes from itself to other routers. At present, the Dijkstra algorithm - implemented in OSPF protocol - is used. The problem is, that this algorithm is sequential from its nature, what is a serious drawback in multicore routers.

Hence, there is a need for a good parallel routing algorithm, which may be applied on computers with shared memory, such as equipped with multicore processors modern routers.

In the paper we will review several classical and new routing/shortest path algorithms[2], such as: Dijkstra, Bellman-Ford, SLF-LLL, auction algorithms, Q-routing and ant(swarm)-routing, paying a special attention to their potential for parallelization. Then, we will describe the results of a series of tests on the problems obtained from GRIDGEN, which is a popular network problems generator. At the end some recommendations will be given.

2. ORIGINAL VECTOR BELLMAN-FORD ROUTING ALGORITHM

We consider a directed graph consisting of n nodes (routers). Let us denote by N the set of all these nodes and by A the set of all arcs in the directed graph (that is, the set of all links in the network). Let us assume, that every arc $(i,j) \in A$ is characterized by a positive scalar value a_{ij}, which we will treat as the cost of passage from i to j, that is the distance measure (metric). In most routing problem statements it is taken $a_{ij} = 1 \; \forall (i,j) \in A$, that is the length of the path is simply the number of links (hops).

Let us fix the source node $s \in N$ and assume, that all other nodes may be reached from it. Denote by $d_i^s \triangleq d_i$ the cost of passage from the node s to a node i and by $\hat{d}_i$ its minimum value. It can be easily proved (e.g., by the contradiction), that the paths of the minimum costs (so-called shortest paths) can be obtained through the solution of the following set of equations:

[2] Routing tables may be calculated by solving shortest-path problems for connections from a given router to every possible destination. In fact, due to Bellman Principle of Optimality, it is not necessary to consider all the destinations.

$$\hat{d}_j = \min_{(i,j)\in A} \left(\hat{d}_i + a_{ij} \right) \quad j \neq s \tag{1}$$

Let us take now

$$h_j(d) = \begin{cases} \min\limits_{(i,j)\in A} (d_i + a_{ij}) & j \neq s \\ 0 & j = s \end{cases} \tag{2}$$

and

$$d = [d_1, d_2, \ldots, d_n]$$
$$h = [h_1, h_2, \ldots, h_n] \tag{3}$$

To find the solution we may apply the Bellman-Ford algorithm

$$d := h(d) \tag{4}$$

starting from $d_s = 0; \; d_j = \infty, j \neq s$. This algorithm is based on the order preserving (monotone) mapping h, which may be implemented in the parallel, totally asynchronous version with the arbitrary division of the vector d (Bertsekas and Tsitsiklis, 1997).

The optimization algorithm (2)-(4) may be applied repetitively - to adapt routing to the current situation in a network. In that case the cost a_{ij} should be a measure of the quality of transmission, dependent on the current flow (transmission rate) f_{ij} between nodes i and j. A very popular flow cost function $a_{ij}(.)$ is:

$$a_{ij}(f_{ij}) = \frac{f_{ij}}{(c_{ij} - f_{ij}) + \varepsilon_{ij}} + \delta_{ij} \cdot f_{ij} \tag{5}$$

where c_{ij} is the transmission capacity of arc (i,j) and δ_{ij} is the processing and propagation delay (of course we assume $0 \leq f_{ij} \leq c_{ij}$; $\varepsilon_{ij} > 0$ is a small constant to avoid zero in the denominator). However, in this - adaptive - case one should remember, that the cost functions (5), dependent monotonically on flow, should be augmented with constant components σ_{ij} (so-called bias factors, interpreted as link costs/lengths at zero load), because otherwise oscillations (in subsequent optimal routings) may occur (Bertsekas and Gallager, 1992).

The presented adaptive routing approach is used in the Internet in demons *routed*, *gated* and protocols RIP and Hello (Comer, 1991). In the first protocol so-called "hop count metrics" is used, what means, that simply all elementary arcs are counted for; in the second - "network delay metrics", that is the time of transmission is taken into account. In the active state, all messages used to the optimization of the routing tables (i.e., the tables of the shortest path neighbors for different destinations) are sent by every computer to all direct neighbors every 30 seconds.