How Many Cores Does Parallel BGP Need in a High-Speed Router

Yaping Liu, Shuo Zhang, Zexin Lu and Baosheng Wang
School of Computer Science, National University of Defense Technology
Changsha, Hunan, P.R.China
e-mail: {ypliu, zhangshuo, lzx, wbs }@nudt.edu.cn

***Abstract*—The performance problem of BGP has raised great concerns both in industry and research. With rapid expansion of Internet, how to improve the performance of BGP to support more BGP neighbors in a high-speed router is a practical urgent problem. In this paper, we presented a Minimal Cores Computing (MCC) algorithm based on multi-root tree model to compute the minimal cores for parallel BGP in the context of the multi-cores platform. The algorithm is an approximation algorithm as the problem is a nonlinear programming problem. Simulation results show that MCC can get reasonable good speedup with minimal number of cores. MCC can give direction to the design of the control node in a core router.**

Keywords-parallel BGP; speedup; multi-cores; performance; router.

I. INTRODUCTION

With rapid expansion of Internet, BGP (Border Gateway Protocol) [1] confronted serious performance problem. Feldmann [2] pointed out that with the increase of BGP neighbors, the router can not deal with the update packets immediately, thus the routing update time will increase. If the neighbors are beyond 250, the router will hardly maintain these neighbors even if it does not process any other packets. According to Agarwal [3]'s analysis, rapid route changes have made BGP process consume over 60% of CPU cycles.

To overcome the low-efficiency of BGP, some mechanisms are used to improve its scalability, such as confederation and route reflector. However, these mechanisms require more complex network configuration, which may lead to configure mistakes. The performance problem also exists.

At present, the main solution is to use BGP distributed computing, which focuses on the corporation among BGP instances on different control nodes in one router. However, with the development of multi-cores CPU, we can create parallel BGP on multi-cores CPU, which focuses on the model and implementation in one node. Comparing with dsitributed BGP, parallel BGP has much lower communication cost. And one multi-cores control node is cheaper than multiple control nodes. Our research belongs to parallel BGP on a multi-cores CPU.

We present a multi-root tree model (MR-PBGP) for parallel BGP, which is an integrated model with neighbor-based division and data division. According to the model, the problem of compute the minimal cores for a BGP router with appointed number of neighbors is a nonlinear programming problem. Thus, we presented an approximation algorithm named MCC (Minimal Cores Computing algorithm) and simulated it using Matlab. Simulation results show that the algorithm can get a sound result giving appointed number of neighbors, the probability of comparing all routing information to select an optimal route, the probability of routing updates with the same prefix from different neighbors, the probability of EBGP, and the performance ratio to the ideal situation or an appointed parallel speedup.

To the best of our knowledge, there is no previous work on determining the minimal cores for parallel BGP similar to ours.

The rest of the paper is organized as follows. Section II reviews the related work. Section III describes our problem and presents MCC algorithm. Section IV simulates MCC in the Matlab environment. Section V draws conclusions for this research.

II. RELATED WORK

The research about BGP protocol parallelism mainly centers on BGP distributed computing. It catches high-end router manufacturers' attentions especially [4-6].

Markus Hidell proposed a distributed BGP protocol model based on data division [7]. Its main idea was to divide all of the network prefixes into several disjoint subsets, and assign different network prefixes to different BGP protocol entities to do the parallel processing. However, its main drawback is that with the increase of neighbors, session management (SM) will become a bottleneck. This model is suitable for the network environment that has relatively small number of neighbors.

Kun Wu et al. [8,9] proposed a distributed BGP route processing model based on a tree structure. Their study was based on the following two assumptions. First, BGP can select the optimal route without the requirement of the whole route information for the same network prefixes. Secondly, they mainly centers on how to improve the performance of the process of optimal route selection of BGP.

Xiaozhe Zhang [10] proposed an agent-based distributed parallel implementation BGP model. The model introduced the team work idea of the agent technology, extended BGP protocol to be a BGP entity independently running on each control node.

Although the above mentioned methods are the parallel processing techniques of protocols, they are not suitable for multi-cores platform.

III. MCC ALGORITHM

To simply the problem, no routing policies are configured. Suppose that we use t_1 to represent the average time in creating a new entry, t_2 to represent the average time in selecting the optimal route for one prefix, t_3 to represent the average time of unpacking the optimal route for one prefix, t_4 to represent the average time of advertising the optimal route for one prefix, if its neighbor is an ebgp neighbor. Moreover, n is the number of neighbors, x is the set of routing prefixes, p_1 is the probability of comparing all routing information to select an optimal route, p_2 is the probability of routing updates with the same prefix from different neighbors, p_3 is the probability of EBGP, and finally, λ is the average arrival rate of routing updates.

A typical serial model of BGP can be considered as a queue system of M/M/1. Its average service time S_o satisfies (1).

$$s_o = t_1 + t_2 + t_3 + p_3 n t_4 \tag{1}$$

Theoretically speaking, t_3 and t_4 are constants. The value of t_1 is related to the size of x. Since the cost of allocating memory is far greater than the cost of searching and inserting in a structure of tree, the value of t_1 mainly depends on the cost of allocating memory, hence it can also be treated as a constant. The value of t_2 is proportional to the number of neighbors, from which a router receives routing updates for one prefix.

Let $k_i = \frac{t_i}{t_1} (i \geq 2)$, then the average service rate satisfies (2). The value of k_2 is a liner function of n. Moreover, k_{2a}, k_{2b}, k_3, k_4, and C are constants.

$$\mu_o = \frac{1}{s_0} = \frac{1}{1 + k_2 + k_3 + p_3 n k_4} \times \frac{1}{t_1}$$

$$k_2 = k_{2a} n + k_{2b} \tag{2}$$

$$C = \frac{1}{\lambda t_1}$$

If $\lambda < \mu_0$, the average stay time of one bgp message in the router is t_s, which, by the queue theory, equals (3).

$$t_s = \frac{1}{\mu_o - \lambda} \tag{3}$$

To decrease the value of t_s, we propose a multi-root tree model and call it MR-PBGP as illustrated in Fig.1. This model can not only reduce the arrival rate but also cut down neighbors for each thread, so that k_2, $p_3 n$, and λ are decreased.

In Fig. 1, *w* represents the number of roots, *h* represents the height of the tree, and x_i represents sons of the ith-level node (not leaf node). Leaf nodes represent bgp neighbors. The first-level threads are nodes that are fathers of leaf nodes. Every first-level thread creates several BGP sessions and handles BGP routing information from its neighbors. Each first-level thread may have multiple fathers depending on *w*. Different father node deals with different scope of routing prefixes showing ideas of data division. If the selection of optimal route for one prefix requires all routing information from every neighbor, the routing information of that prefix will be sent directly to the corresponding master-thread by the first-level thread. Thus, in Fig.1, there are direct lines from the first-level nodes to root nodes.

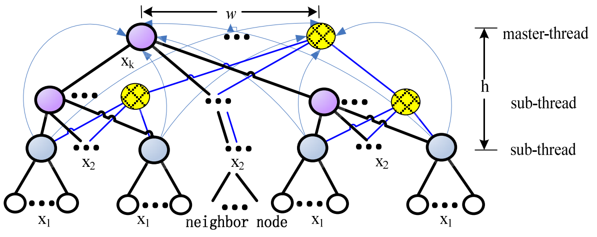

Figure 1. MR-PBGP

The problem of compute the minimal cores according to MR-PBGP can be described as following (4).

$$\lambda_0 = \lambda \times \frac{1}{n}$$

$$\lambda_1 = \lambda \times \frac{x_1}{n} = \lambda_0 \times 1 \times x_1$$

$$s_1 = (1 + k_3 + (1 - p_1 p_2) k_2 (x_1) + p_3 x_1 k_4) t_1$$

$$\mu_1 = \frac{1}{s_1}$$

$$t_{s1} = \frac{1}{\mu_1 - \lambda_1}$$

$$\lambda_2 = \frac{1}{w} \times \frac{\lambda_1}{x_1} \times (\frac{p_2 (1 - p_1)(x_1 + 1)}{2} + (1 - p_2) \times x_1) \times x_2$$

$$s_2 = (1 + k_2 (x_2)) \times t_1$$

$$\mu_2 = \frac{1}{s_2}$$

$$t_{s2} = \frac{1}{\mu_2 - \lambda_2}$$

……

$$\lambda_k = \frac{\lambda_{k-1}}{x_{k-1}} \times (\frac{p_2 - p_1 p_2}{1 - p_1 p_2} \frac{x_{k-1} + 1}{2} + \frac{1 - p_2}{1 - p_1 p_2} \times x_{k-1}) \times x_k$$

$$s_k = (1 + k_2 (x_k)) \times t_1$$

$$\mu_k = \frac{1}{s_k}$$

$$t_{sk} = \frac{1}{\mu_k - \lambda_k} \qquad (k<h)$$

$$\lambda_h = \frac{\lambda_{k-1}}{x_{k-1}} \times (\frac{p_2 - p_1 p_2}{1 - p_1 p_2} \frac{x_{k-1} + 1}{2} + \frac{1 - p_2}{1 - p_1 p_2} \times x_{k-1}) \times x_h + \frac{1}{w} \lambda p_1 p_2$$

$$s_h = [\ 1 + (1 - p_1 p_2) \times k_2 (x_h) + p_1 p_2 \times k_2 (\frac{n}{2})\ ] \times t_1$$

$$\mu_h = \frac{1}{s_h}$$

$$t_{sh} = \frac{1}{\mu_h - \lambda_h}$$

$$\lambda_{ideal} = \lambda \times \frac{V}{n}$$

$$s_{ideal} = (1 + k_3 + (1 - p_1 p_2) k_2(V) + p_3 V k_4) t_1$$

$$\mu_{ideal} = \frac{1}{s_{ideal}}$$

$$t_{ideal} = \frac{1}{\mu_{ideal} - \lambda_{ideal}}$$

$$\min_{x_1, x_2, \ldots, x_h} m = w \times (1 + \sum_{k=3}^{h} \prod_{i=k}^{h} x_i) + \prod_{i=2}^{h} x_i \qquad (4)$$

Subject to:

$$\max(t_{s1}, t_{s2}, \ldots, t_{sh}) = A t_{ideal}, A > 1 \text{ is a constant}$$

$$x_1 \times x_2 \times \ldots \times x_h = n$$

$$x_1, x_2 \ldots, x_k \geq 2, w \geq 1, h \geq 2 \text{ are integers}$$

```
Algorithm: MCC
Inputs: $p_1,p_2,p_3,C,k_{2a},k_{2b},k_3,k_4,n,V,A$; Outputs: $m$, $x_i$, $j$, and $w$
1) Solve the following equation and set the value of $x_1$
      $t_{s1} = At_{ideal}, x_1 = \lceil result \rceil$
2) Compute the possible max height of the tree $H$
      $H = \log_2^{n/x_1} + 1$
3) $j$=2; /*search the optimal values*/
   while (j < H) do
      if (j= =2 ) then
         $x_2 = \lceil n / x_1 \rceil$;calculate $w$ according to $t_{s1} = t_{sh}$;
         if ($w$<1) then
            break;
         end if
         else then
            Calculate $m$ according to (4); $j$++;
            continue;
         end else
      end if
      else then
        optimizer($w$,$j$, $x_i$);
      end else
   end while
                        optimizer($w$,$j$, $x_i$)
   $w$--;
   if ($w$ < 1) then
      return;
   end if
   else then
      compute the optimal integer values of $x_i(i=2,\ldots)$
      according to $t_{s1} = t_{s2} = \ldots = t_{sh}$;
      if (no feasible solutions) then
         $j$++; return;
      end if
      else then
         calculate $m$ according to (4) and update $w$,$j$, $x_i$;
         optimizer($w$,$j$, $x_i$);
      end else
   end else
```

Figure 2. Pseudo Codes of MCC

In formula (4), *V* is the value of the number of the first-level thread's neighbors calculated by the ideal MR-PBGP model with no constraint on cores. It can be proved that the ideal MR-PBGP model is a multi-root binary tree with *V* being greater than 2 [11]. The value of *V* can be calculated by the following (5), in which *v* is the minimal value of x_1 according to $s_h \leq t_{s1}$.

$$V = \max(\lceil v \rceil, 2) \qquad (5)$$

And usually, the value of *V* is equal to 2. The above problem is a nonlinear programming. Thus, we presents an approximation algorithm named MCC as illustrated in Fig. 2.

The complexity of MCC depends on the cost of solving equations and searching times. We use Newton iterative method to solve polynomial equations. Its complexity is O(*zM*), in which *M* is max iterative number and *z* is highest-degree of a polynomial. In MCC, the highest-degree of equations is 5, so that the complexity of solving equations is O(*M*). The max searching times satisfies following (6), in which *w* is the solution of equations with j=2 in MCC.

$$\sum_{j=2}^{H} w(j-1) = \frac{H(H-1)w}{2} < w(\log_2^n)^2 \qquad (6)$$

As *w* is considered to be a constant in this case, the complexity of MCC is O((log$_2$n)^{2}M).

IV. SIMULATION

We simulated MCC using MatLab. We use Quagga BGP [12] as the sample of typical BGP running on Lenovo with 4-cores Intel Xeon E5405 CPU, 4G Memory, and two 1G Ethernet interface. Tested by Spirent AX4000 [13], C, k_2, k_3 and k_4 were obtained as follows:

$$C = 2.93, k_3 = 0.32, k_4 = 0.15$$

$$k_2 = k_{2a} \times x + k_{2b} (k_{2a} = 0.03243, k_{2b} = 0.58122)$$

(x is neighbors)

At first, we chose p_1 as 0.1, p_2 as 0.7, p_3 as 0.2, which is nearer to current real network [14]. The neighbors change from 64 to 2048. As the serial typical BGP do not accord with conditions of queue theory in those cases, parallel speedup should be infinite. Thus, we chose performance ratio with that of the ideal parallel model changing from 0.1 to 0.7.

Fig. 3 shows the simulation results of cores computed by MCC. It shows that the number of cores is increased with the increase of neighbors and performance ratio. The cores change from 3 to 10 with neighbors changing from 64 to 2048 in the case of performance ratio equaling 0.1. And the cores change from 9 to 42 in the case of performance ratio equaling 0.7. The increase speed of cores is higher with high performance ratio than that with low performance ratio. The phenomena also show that if a router wants to support large neighbors with performance near to the ideal optimal value, a large number of cores are needed. However, if a router wants to support large neighbors with acceptable performance, only several cores are needed. For example, if we chose performance ratio to 0.1, only 10 cores are needed to support 2048 BGP neighbors with its parallel speedup being infinite.

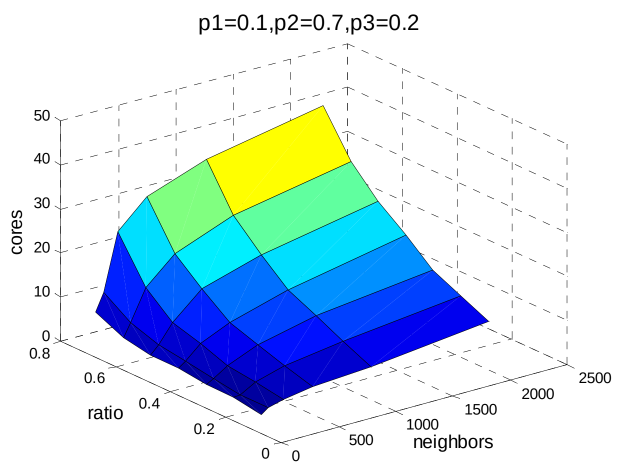

Figure 3. Cores computed by MCC

Fig. 4 shows the simulation results of *w* computed by MCC. The value of *w* represents the width of MR-PBGP tree. The results show that if the value of performance ratio is lower, *w* often equals to 1 which means one master thread in the model. With the increase of performance ratio, *w* also increases. The reason is that the cost time of first-level thread becomes lower so that the master thread may be bottleneck. Thus, *w* should increase to reach load balance between threads. In our simulation, the value of *w* does not be beyond 4 in most cases.

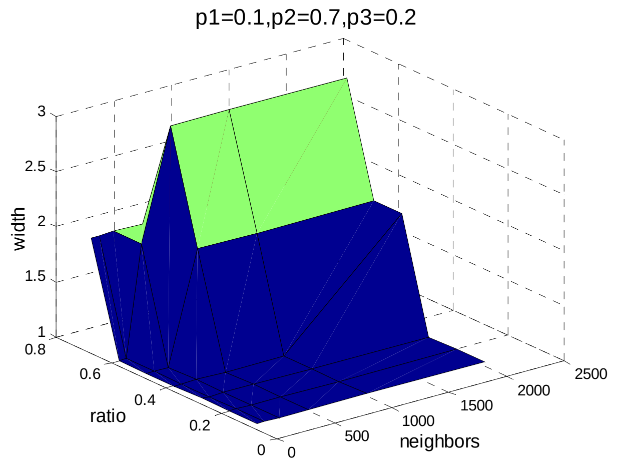

Figure 4. w computed by MCC

Fig. 5 shows the simulation results of *h* computed by MCC. The value of *h* represents the height of MR-PBGP tree. The results show that in most cases, the height of the tree in the model is 2. And with the increase of performance ratio and neighbors, *h* may increase. The reason is that the cost time of master thread can decrease more rapidly by increase of *w* than by increase of *h*. Thus, the value of *h* becomes high only with high performance ratio and large neighbors. The value of *h* usually does not be beyond 4.

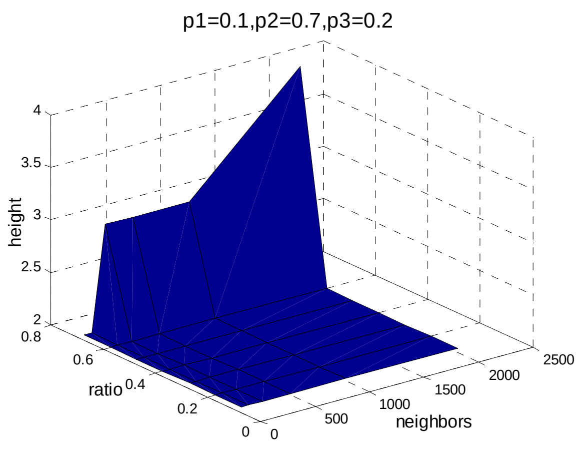

Figure 5. h computed by MCC

Then we research on the relationship among cores, p_1, p_2, and p_3 giving appointed neighbors and performance ratio. We chose neighbors as 2048, and performance ratio as 0.1.

Fig. 6 represents the simulation results about the change of cores, *w* and *h* with the change of p_1. The result shows that the cores, width and height change little with the increase of p_1 excluding p_1 equaling 1. The value of core number is around 10. The value of *h* is 2 or 3. And in most cases, the value of *w* is 1. The reason is that though the value of x_1 increases with the increase of p_1, the bottleneck of system mainly lies on the first-level thread in most cases. Thus, the core number changes little. If p_1 is equal to zero, the value of *V* (according to (5)) is equal to 2. But if p_1 is not zero, *V* increases quickly (For example, if p_1 is equal to 0.1, *V reaches 34*). The value of x_1 computed by MCC is much lower in the case of p_1 equal to zero. But for higher values of p_1, the bottleneck of the system is master thread *so that w* increases rapidly which leads to cores' increasing rapidly. Thus, the value of cores is firstly slightly decreased and increased for higher values of p_1.

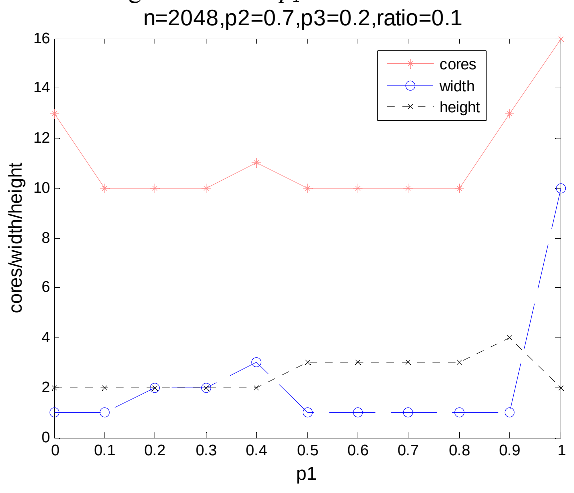

Figure 6. m/w/h with p1

Fig. 7 represents the simulation results about the change of cores, *w* and *h* with the change of p_2. The results show that the width (*w*=1) and height (*h*=2) keep constant with the increase of p_2. The value of core number decreases with the increase of p_2.The reason is that the cost of master thread decreases with the increase of p_2. We can decrease cores by increasing the value of x_1.

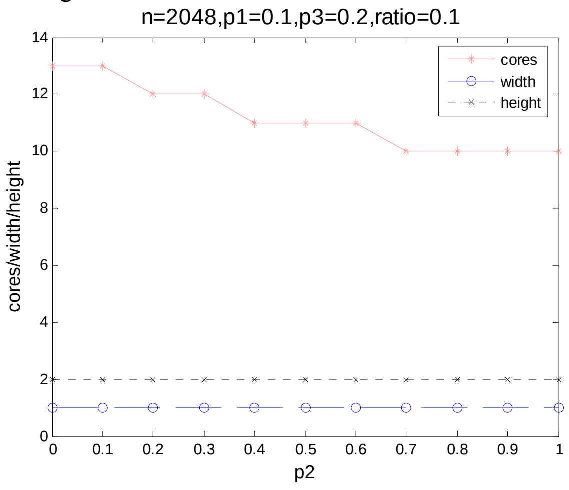

Figure 7. m/w/h with p_2

Fig. 8 represents the simulation results about the change of cores, *w* and *h* with the change of p_3. The results show that the width (*w*=1) and height (*h*=2) keep constant with the increase of p_3. The value of core number increases with the increase of p_3.The reason is that the cost of first-level thread increases with the increase of p_3. To reach load balance, the value of x_1 should be decreased so that cores increase.

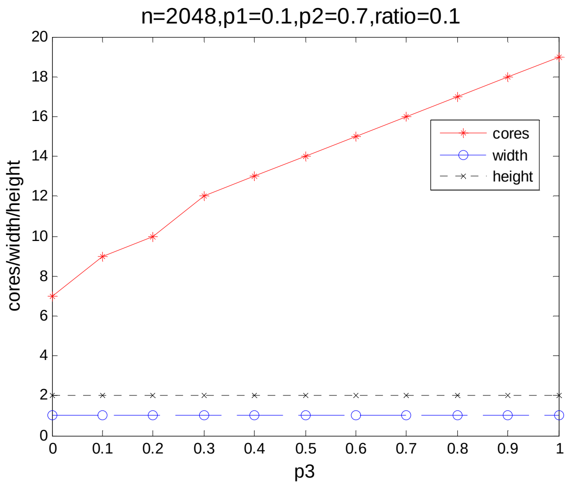

Figure 8. m/w/h with p_3

In a word, if we chose an acceptable low value of performance ratio, the core number computed by MCC is small with the condition of supporting large neighbors. The current common high-end multi-cores CPU can satisfy the condition. For example, 8-cores CPU can support 2048 BGP neighbors with acceptable parallel speedup. Moreover, the tree is two- level structure in this situation with *w* equaling to1 in most cases. However, if a router wants to be designed to support large neighbors with its parallel speedup near to its ideal value, a large number of cores are needed. In this situation, the current highest-performance CPU are needed with 64-cores or 128-cores. And the model will appear as a multi-root and multi-level tree.

If we want to implement parallel BGP in a high-end router, there are three steps to do it. At first, we need testing the values of C and k_i according to the supported interface rate, CPU, and original implementation of typical serial BGP. Secondly, we can determine the cores by algorithm MCC with requirement of supporting number of neighbors, and other parameters. At last, we can implement parallel BGP with multiple threads according to the model of MR-PBGP and the parameters's values computed by MCC. It is our current work to implement a real prototype.

V. CONCLUSION AND FUTURE WORKS

Our main contribution of this paper is presenting a minimal cores computing algorithm named MCC to compute the minimal cores for parallel BGP oriented the multi-cores platform. Simulation results showed that current common high-end multi-cores CPU can be used in a high-speed router with an acceptable good speedup supporting large BGP neighbors. The model and the algorithm are very usable to the design of high-speed routers. However, in real networks, BGP may get dynamic payload from different neighbors. We are implementing a real prototype for parallel BGP in a high-end router. We will test the performance of the prototype and research on how to reach the optimal status under a dynamic running environment.

ACKNOWLEDGMENT

This work is supported by the National Natural Science Foundation of China under Grant No. 61070200, the National Key Technology R&D Program under Grant No. 2008BAH37B03-03, and the National High-Tech Research and Development Plan of China under Grant No.2008AA01A325.

REFERENCES

[1] Y. Rekhter. A Border Gateway Protocol 4 (BGP-4), 2006. RFC 4271.

[2] Feldmann, H.W. Kong, O. Maennel, and A. Tudor. Measuring bgp pass-through times. Barakat C and Pratt I, editors, In Passive Active Measurement Workshop, vol. 3015, pp. 267-277. France: Springer-Verlag GmbH, 2004.

[3] S. Agarwal, C.N. Chuah, S. Bhattacharyya, and C. Diot. Impact of bgp dynamics on router cpu utilization. Barakat C and Pratt I, editors, In Passive Active Measurement Workshop, volume 3015, pp. 278-288. France: Springer-Verlag GmbH, 2004.

[4] D. A. Ball, R. E. Bennett, et al. Distributed software architecture for implementing BGP, Patent Application Publication, no. US 2005/0074003, April, 2005.

[5] Juniper Networks, Inc. T640 routing node and TX matrixTM platform: architecture. White Paper (Part Number 350031-001), http://www.juniper.net, Dec, 2004.

[6] Cisco Systems, Inc. Next generation networks and the cisco carrier routing system. White Paper, http://www.cisco.com, Dec, 2004.

[7] Markus Hidell, Peter Sjödin, Tomas Klockar, and Lenka Carr-Motyckova. A Modularized Control Plane for BGP, IEEE Proceeding PDCS07, 2007.

[8] K. Wu, J. Wu, and K. Xu. A tree-based distributed model for BGP route processing. In: HPCC 2006, LNCS 4208, Berlin: Springer-Verlag, pp. 119-128, 2006.

[9] K. Xu and H. He. BGP parallel computing model based on the iteration tree. Journal of China Universities of Posts and Telcommunications, 15(Suppl.), pp. 1-8, September, 2008.

[10] X. Zhang, P. Zhu, and X. Lu. Fully-distributed and highly-parallelized implement- tation model of bgp4 based on clustered routers. In: ICN 2005, LNCS 3421, Berlin: Springer-Verlag, pp. 433-441, 2005.

[11] Y. Liu, S. Zhang, G. Zhang. Parallel BGP: Analysis, Model and Experiment. Technical Report, School of Computer Science, National University of Defense Technology, Sep, 2011.

[12] Quagga project. Available from http://ww.quagga.net/download/, Feb, 2011.

[13] Spirent Adtech AX4000 broadband test system. Available from http://www. smartechconsulting.com/Adtech-Spirent-AX4000-AX-4000-16-Slot-Rack-mount-Chassis, Oct, 2010.

[14] CIDR Report [EB/ OL] [2008-2-10]. http :/ / www. cidr-report . org, Sep, 2009.