IJREAT International Journal of Research in Engineering & Advanced Technology, Volume 1, Issue 2, April-May, 2013
ISSN: 2320 - 8791
www.ijreat.org

Reduce Congestion Control using Virtual Key Identifier

S.PRAVEENA[1], S.JAYANTHI[2], S.LALITHA[3] and G.PRIYADHARSINI[4]

[1] Asst. Professor, Department of Information Technology, Adhiparasakthi Engineering College, Melmaruvathur-603319, Tamil Nadu, India.

[2,3,4] 3rd Year Department of Information Technology, Adhiparasakthi Engineering College, Melmaruvathur-603319, Tamil Nadu, India.

Abstract

The congestion control problem in combination of distributed and peer to peer networks through the distributed hash table is dealt in this paper. The main goal of DHT network for mapping identifier to key address and select the socket address and also it makes high lookup rates. In addition, it efficiently makes data flow for heterogeneous network. Most of the routing algorithms are effectively transfer the user requests without congestion. Sometimes node gets overloaded due to the network heterogeneity at same time DHT networks have load balancing problem. In existing (ERT) mechanism we avoid the query load balancing using Degree calculation technique, but the main constraint of this scheme does not support large geographical area. In this paper, we proposed priority based virtual identifier. This identifier dynamically constructs the virtual tables and it does create virtual key identifier. Based on the priority algorithm, key identifier points the high degree node and reduces the congestion. This virtual table can also provide the confidentiality data transaction**.**

Keywords: *Area based Congestion Identifier, Backward queue Transmitting*

1. Introduction

Thenetwork connectivity is classified into two areas. They are interconnected and intra connected. The parallel processor or some cluster based system support for interconnected network. The interconnection network or parallel processing networks always contains the key element for storage purpose. Based on the key elements we store the multiple numbers of data's, so that automatically the size of network will increase. Sometimes congestion will arise based on the topology setup.Simple networks follow chord topology. Most of the simulation results come under the chord topology. Toreduce the complexity, time constraints and power consumption, it takes long term process of interconnection.At the same time developer should transaction of interconnected networks getting higher rate of traffic. At the time, traffic reduction process alsocontains high cost process. This type of network congestion will also affect those who all are connecting with other subnetworks. This congestion will associate with the traffic jam. This type of traffic jam will also affect the internal path. All internet subsystems having simultaneous request and response sequence includes the packets transaction. Most of the internal and external factors affect the network packet loss. In this situation multiple number of input packets will wait for some other external packet response until queue will be free.

Normally the parallel processor has very large size and multiple numbers of interconnected systems, the more number of interconnecting systems are criticalto handle the working space. Monitoring the subsystem is very critical one. The future trend in every node is considered by virtually or multiple number of subsystems. The virtual connections reduce the construction cost and power, and geographical area space. But the interconnected nodes did not reduce the congestion. This congestion mainly affect the network work performance so, the network congestion reduction is act a virtual role for every network subsystems. Area based congestion identifier (ABCI) can point where congestion will occur. Based on this notification we can cure the congestion by theoretically. After identification of the ABCI, the whole setup transfer to some priority based queues. After the priority based consideration we can remove one by one or simultaneously.

Suppose this priority based congestion reduction take more processing time, another method we can use the redirection of the congestion of one network end system to other network end system. The network redirection is 100% supported by distributed management networks. This will not take the intermediate memory space. But

the redirection sometimes takes time consumption. To, avoid the above external factors, each network end systems construct some linear table for how much of indegree and outdegree were tie up with each other. These tables prevent the congestion multiple number of subsystems. Apart from the routing table this is considered only for end level subsystem with connection and load details and the content of messages. So, this is time consuming process in distributed networks.

In modern networks, some sensors based tools can reduce the congestion. For the fast dispatching purpose only this type of tools will be added. But sometimes these tools get failure. So, we dealt with this problem to connect one virtual table or control based table indicate the incoming and outgoing congestion details. If one queue fills the packet or it could not transfer the load, that time "**backward queue transmitting**" will reduce the congestion. So, each congestion reduction queues forward and backward transmitting. The queue can be split into various number of tables based on the traffic that will load the traffic details. Based on the high network performance reduce the average waiting time increase the network throughput.

Based on the network area we can identify forward queue and backward queue. Where the congestion will always occur we can create double type queue that is forward queue and backward queue. That queue will connect with virtual tables. All queues always involve either forward queue or backward queue. If congestion affects the $1/3^{rd}$ size of the network, automatically queue will increase the size based on the congestion table for security purpose and the queue will be blocked. And the content will cover the security codes. **"status identification header (SIH)**" may get the detail of backward queue status. This SIH may prevent the network traffic packet flows.SIH is the header used to block the details and attach the next possible available space or port. This acts like advanced switches. This whole setup embed with work on the ABCI(Area based congestion identifier).ABCI's working principle is completely based on the forward and backward queue in order to reduce the congestion ABCI which is to identify the heavy traffic area and immediately construct the queues as dynamically, and it isolate the next possible paths.

2. Queue based routing for Multiple Connected Paths (MCP)

The output port is computed by queue based routing in which it checks a few bits in the Destination ID of the packets along its routes. It does not require forwarding tables for the implementation of routing. Let us consider, for an m-stage of n*n switches in which T is a series of m routing steps (T=tn-1…..t1t0), where the output port at stage i is represented as the value for routing the step ti.Thus the queue based routing calculation is given as ti=dj, where i represents a MCP stage, and j depends on the destination of MCP topology.
In single route MCPs, only single path is laid between the source and destination. Thus a single queue is estimated for each

1. To specify each type in MCPs, we can refer both the unidirectional and bidirectional MCPs (BMCPs).
2. The number of input and output is represented as p.

In BMCPs, "Upward path survive in each source/destination pair; if the "turnaround connection is established then there is single "downward" path. The deterministic routing is calculated for both the source/destination pair.

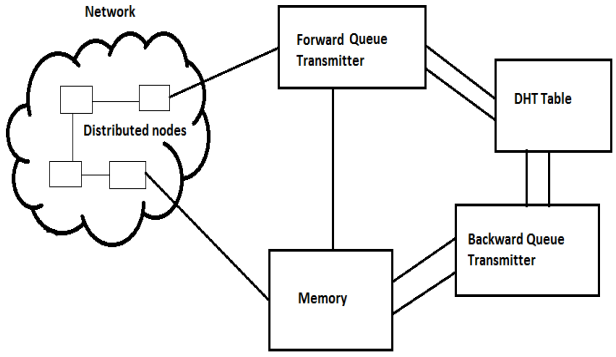

Fig. 1 Queue Based Routing

If congestion occurs on the destination, DET (Dynamic Routing Algorithm) is one in which each switch is used to select the output ports. Packets are received at last stage to address the destination and forwarded to a specific downward path to address another destination. Queue based routing is a type of algorithm in which each link is assigned to destination and the bits are

common to this destination. Significant bits are selected by output ports. Least significant bits at stage 0, Second-least significant bits at stage 1, most significant bit at stage 2.

In a k-ary n-tree is a straight forward direction of fat-trees they are built from switches of constant radix in which it has P*P bidirectional switches that has a destination D=dn-1….d1d0 (0<=di<=k-1). The formula for queue based routing for DET is ti=di. The turnaround stage is calculated by using source and destination packet. Let us consider a source packet is 111 and destination packet is 010 has routing tag 010 (i.e. routing steps are: t2=d2=0; t1=d1=1; t0=d0=0) and has a turnaround 2.

Adaptive routing performance is achieved by balancing the network traffic. The same numbers of paths are addressed by all links of a given stage.

2.1 Input port organizations for Dynamic Based Congestion Controller (DBCC):

The turnaround connections are used for selecting the output port for crossing both the upward and downward packets. BMCPs uses similar routing algorithm and queue based implementation. DET are efficient for the above purpose. Degradation of network cannot be avoided during congestion situations. The effective bisection bandwidth offer an Infinite band based on MCPs that leads to network congestion. Thus queue based routing is implemented to overcome the problem in congestion occurrence.

3. DBCC Description

Congestion deals with basic ABCI principles that follow DBCC. It introduces innovations regarding the encoding of the location of congestion roots, the structures for storing congestion information, the identification of forward queue transmitting and the propagation of congestion information which differs in essential aspects of ABCI-like Solutions. The queue based routing is implemented using the network in a MCP with a DET-like routing algorithm. It reduces the number of RAM and control memory to implement DBCC.

3.1 Switch Architecture

The Manufacturers of high performance interconnects are preferred only by input ports in which switches with RAMs are used. The RAM at input ports has two types to manage queues:

1. Backward queue transmitting (BQT: One per input port)
   It forwards packets regardless to their destination.
2. Forward queue transmitting (FQT: a set of queues per input port)
   It forwards packet to source are isolated. It is widely used in real clusters, DBCC).

The control memory at input ports is implemented with an Addressable Memory. Each Memory contains the information about the forward queue transmitting and to specify the congestion root. Memories are also required at output ports to propagate congestion information between switches.

.

3.2 Congestion Detection and Location

Like ABCI-like solutions, by measuring the occupancy level of the Backward Queue Transmitting (BQT) it detects the congestion root using DBCC. At some output port of the switch, the congestion root has appeared if the level exceeds the given detection threshold. The congestion root is located at output port for contributing the “overfilled” BQT which assumes that the packet is at the head by DBCC. The destination is labeled in each packet. The queue based routing is used to compute the routing logic. The packet at the head of the queue describes where detection occurs and is not contributing to congestion, the failure of post-processing mechanisms are solving in later sections.
The number of depicted ports is not restrictive as DBCC does not limit the number of switch ports. Each packet is requested by its output port. , i.e. OP = tstg (dest), where “dest” is the destination of the packet and “stg” is the network stage where the switch is placed. Thus, tstg (dest) indicates the routing step of that stage computed for the destination. Specifically, we assume that the routing step of that stage is computed from the two least-significant bits of the destination.
The value of the Detection threshold is a key parameter of DBCC that should be carefully tuned, in order to obtain the best DBCC performance.

3.3 Congestion Data Encoding and Storage

To encode the location of any detected congestion, root is a criterion of the key aspect of DBCC. By using this information, we can identify the hot packets contributing in the congestion root. The same output port is assigned to packet destination in which DBCC assumes queue based DET-like routing. From this point of view of "in advance" detecting packets that will pass through a specific output port, which port can indicated as the value of the bits that the destinations assigned to it have in common.

The corresponding mask is computed and stored in the mask field and it detects the congestion root located in the switch. Congestion is detected by memory line at the input port. Memory lines could be allocated at the same input port to store different masks, in order to keep track of other congestion roots. The destination of any packet in a port must be compared with all the masks stored in the "active" memory lines at that port.

3.4 Forward Packets Isolation

Once a memory line has been allocated at an input port to be aware of a congestion root, an FQT is also dynamically allocated at that input port to isolate packets contributing to that root. BQT is mapped initially for reaching all the packets at an input port. Once a memory line has been allocated at an input port to be aware of a congestion root, an FQT is also dynamically allocated at that input port to isolate packets contributing to that root.

In the case of a match, the packet is detected as a hot one and then moved to the back of the corresponding FQT; otherwise the packet remains in the BQT, ready to be scheduled and forwarded. Hence, hot packets are immediately removed from the head of the BQT and separated from the cold ones. Moving packets from the BQT to an FQT does not mean to move information inside the RAM but to rearrange the pointers controlling the queues. This is a "post-processing" mechanism, as destinations are not processed until packets reach the head of the BQT.

The resulting mask depends not only on the root position but also on network topology. Same destination could be forwarded from two queues of that input port, may be in an order different than they arrived at that port. By contrast, the post-processing mechanism in-order moves the hot packets from the BQT to the FQT, thereby preventing out-of-order delivery that is unacceptable to many applications.

When the packets at the head of the BQT is not actually contributing to congestion, if BQT detects congestion then the post-processing turns harmless. An FQT would be unnecessarily allocated to store packets that are not actually contributing to congestion, and then the packet at the head of the BQT would be moved to the FQT. There would not be a strong contention in using the requested the output port, when the packet was really a cold one.
The packet would be forwarded soon from the FQT without any real penalty. "Wrongly allocated" FQT will become empty rapidly and then eventually reallocated along with its associated memory.

3.5 Congestion Data Propagation

The congestion root is placed only to prevent low – order HoL-blocking when hot packets in the switch are isolated. Hot flows must be identified and isolated at "upstream" switches in order to prevent higher order HoL-blocking. The root is detected when information about the location of any congestion root must be propagated upstream from the switch. The first time that the occupancy of an FQT exceeds a given (Stop) threshold, an "Allocate" notification containing information extracted from the memory line associated to that FQT is sent to the upstream switch.

The input port of memory is stored by using mask. If the "Propagated" bit of the T-memory is set to true to indicate that congestion information has been communicated upstream. A memory line will be allocated at the receiver output port to store the congestion information included in this notification. DBCC focuses on IQ switches, no FQTs can be allocated at output ports, but hot packets can be detected there by comparing their destinations with the masks stored in the memory.

The congestion information from an output port memory line to the input ports of the same switch forwarding hot packets towards this output port which allows propagating upstream. Input ports receiving an

allocate notification from an output port will allocate a memory line as well as an FQT. Input ports are placed one routing step further than the output port with respect to the congestion root, the mask of the newly allocated memory lines must be recomputed to address correctly that root. The mask communicated from the downstream memory line, the number of the sender output port: To obtain the new mask, this number is put in the digits of the mask corresponding to the bits used at this network to compute the routing step.

But the properties of the queue based DET-like routing algorithm, the destinations of packets that can reach the root from the input port is common to the value of all relevant digits of the resulting mask. If input and output ports both in the upward and in the downward directions then the network is a BMCP. The allocated notification is received when memory line and an FQT have been allocated at an input port.

In order to detect and isolate hot flows all along their paths, thereby the HoL-blocking these flows could produce being prevented completely in which congestion information can be propagated upstream, port-to-port. Propagation moves away from the congestion root according to the number of relevant digits of the mask accordingly increases. Allocate notification also includes the entry number of the memory line "generating" it, and this number is stored in the "Next Line" field of the memory line allocated subsequently. This "links" the memory lines addressing the same congestion root in consecutive ports, which is useful for flow control.

3.6 DHT Table Construction in Virtual Network

Normally the parallel processors consider very large size and multiple number of inter connected systems. The most number of inter connecting systems critical to handleworking space. The monitoring the subsystem is very critical one. The future trend is every node is considered by virtually or multiple number of subsystems. The virtual connection reduces the construction cost and power, and geographically area space. But the interconnected nodes did not reduce the congestion. This congestion mainly affect the network work performance so, the network congestion reduction act a virtual role for every network subsystems. Area based congestion identifier (ABCI) can point where congestion will occur. Based on this notification we can cure the congestion by theoretically. After identification of the ABCI, the whole setup transfer to the some priority based queues. After the priority based consideration we can remove one by one or simultaneously.

3.7 DHT Queue Transmitting Algorithm

Choose possible options Node N = {a_1, a_2…} from the outgoing degree set

Consider over load node A
ifsystem has a node Na_1then
Randomly choose a node N_{b1} from N
Else if
Randomly choose both the nodes Na_1 and Nb_1 from N
end if

Choose node Na_1 and Nb_1 for load status
forward R and A to the least heavily loaded node
If Na_1 and Nb_1 get overload
Add to queue
Else if one node has fewer loads another node has heavy load
add heavy node to Backward Queue and
add less node to forward Queue
end if
else
choose nearest node of the subnet

Hash Table	Indegree & Outdegree		Virtual Identifier
	Indegree	Outdegree	
ABC	5	5	
JKL	6	3	
PQR	14	5	14
XYZ	3	12	

Table.1 DHT Table

4. Conclusion & Future Work

We have presented a distributed randomized scheme that continuously rebalances the lengths of intervals of a Distributed Hash Table based on a ring topology. We proved that the scheme works with high probability and that its cost measured the number of migrated nodes is comparable to the best possible. Our scheme still has some deficiencies. The constant analysis is very large considered in the previous systems. We are convinced that these constants are much smaller than their boundsconfiguration yields the best solution. First we will find the parameter is how many approximate nodes are participated in network. Another is how many times a help-offers are forwarded before it is redundant. implied by the analysis. In the experimental evaluation one can play with at least a few parameters to see which The last one has the possibility to describe the lengths of small, medium and large intervals. In future we plan to redesign the scheme so that we can approach the smoothness of 1+_with additional cost of 1/_per operation as it is done. Another drawback at the moment is that the analysis demands that the algorithm is harmonized. This can most likely be avoided with more careful analysis in the part where nodes with small intervals decide to continue or aid. On the one hand, if a node tries to help, it blocks its predecessor for (log n) rounds. On the other, only onedecision is needed per (log n) steps. Another issue omitted here is counting of nodes. Due to the space limitations we have decided to use the scheme proposed by Awerbuch and Scheideler in [3]. We developed another algorithm which is more compatible to our load balancing scheme. It inserts log n markers per node and instead of evening the lengths of intervals it evens the weights defined as the number of markers contained in an interval. We can prove that such scheme also rebalances the whole system in constant number of round.

References

[1]HaiyingShen, Member, IEEE, and Cheng-ZhongXu, Senior Member, IEEE
Elastic Routing Table with Provable Performance for Congestion Control in DHT Networks
IEEE Transactions On Parallel And Distributed Systems, Vol. 21, No. 2, February 2010

[2] M. Adler, E. Halperin, R. Karp, and V. Vazirani.
A stochastic process on the hypercube with applications to peer-to-peer networks.In Proc.Of the 35th ACM Symp.on Theory of Computing
(STOC), pages 575–584, June 2003.

[3] B. Awerbuch and C. Scheideler. Group spreading: A protocol for provably secure distributed name service. In Proc. of the 31st Int. Colloquium on Automata, Languages, and Programming
(ICALP), pages 183–195, July 2004.

[4] J. Byers, J. Considine, and M. Mitzenmacher. Simple load balancing for distributed hash tables. In 2nd International Workshop on Peer-to-Peer

Systems (IPTPS), pages 80–87, Feb. 2003.
[5] B. Godfrey, K. Lakshminarayanan, S. Surana, R. Karp, and I. Stoica. Load balancing in dynamic structured p2p systems. In 23rd Conference of
the IEEE Communications Society (INFOCOM), Mar. 2004.

[6] K. Hildrum, J. D. Kubiatowicz, S. Rao, and B. Y. Zhao. Distributed object location in a dynamic network. In Proc. of the 14th ACM Symp.on Parallel Algorithms and Architectures (SPAA), pages 41–52, Aug. 2002.
[7] D. R. Karger, E. Lehman, T. Leighton, M. Levine, D. Lewin, and R. Panigrahy. Consistent hashing and random trees: Distributed caching protocols for relieving hot spots on the world