DFC: Towards Effective Feedback Flow Management for Datacenters

Baohua Yang*‡, Guodong Li†‡, Yaxuan Qi*‡, Yibo Xue‡§ and Jun Li‡§
*Department of Automation, Tsinghua University, Beijing, China
†Department of Computer Science, Tsinghua University, Beijing, China
‡Research Institute of Information Technology, Tsinghua University, Beijing, China
§ Tsinghua National Lab for Information Science and Technology, Beijing, China
{ybh07, lgd07}@mails.tsinghua.edu.cn, {yaxuan, yiboxue, junl}@tsinghua.edu.cn

Abstract—**Quality of Service (QoS) is essential to datacenters, which requires effective schemes to manage the flows among numerous servers. However, existing management techniques mainly focus on the QoS issues over Internet. This paper presents a Dynamic Feedback Control flow management model for datacenters, called DFC, to achieve performance improvement and fault-tolerance. Based on the new model, mechanisms derived from feedback control are designed and evaluated. By observing system status of each server in the datacenter, DFC successfully applies a smart resources distribution policy and resiles when node breaks down. Preliminary experiments are carried out to validate the effectiveness of the model, based on both the NS2 simulation platform and a datacenter prototype. Evaluation results demonstrate that DFC can effectively improve datacenter's performance and guarantee a good reliability.**

I. Introduction

In the last decade, there is an increasing trend to migrate more and more computing and storage services into very large size datacenters, such as those of IBM [1] and Amazon [2] which consist of huge number of servers. For instance, the datacenter in Google Inc. combines more than 15,000 commodity-class PCs [3] together to provide worldwide search services. One web search request using Google may access thousands of Google Web Servers (GWSs) to inquire petabytes of data storage, and spread heavy flows inside the datacenters. Those heavy flows often lead to unavoidable breakdown of the commodity servers, and some specialized links within datacenters also observe higher loss ratio than others [4]. This implies that the datacenter's performance can benefit from a management of flows, and thus put forward a significant requirement for an effective flow management technology to improve the performance and reliability of datacenters.

Technologies of flow management over IP based Internet have been studied for years [5]–[7]. Most of those works are designed to guarantee the Quality of Service (QoS) of links, such as latency for a specialized video meeting application, or bandwidth for web browsers. Nevertheless, even with decades long effort, an effective and precise flow management over dynamic network system is still lacking, and thus researchers still must confront with that.

Main challenges on flow management over Internet consist of two aspects: reasonable model and available experimental platform. First, current Internet is based on "distributed management" [8], hence a central administration is impractical to be deployed with the present Internet. At the same time, real network systems are dynamic with various types of traffic and different types of user behaviors. Thus it is difficult to propose an accurate model. Secondly, Internet scale experimental platform is hard to be designed and implemented. Though test bed such as PlanetLab [9] provide a global platform for deploying and evaluating network services [10], [11], there are still several deficiencies. For example, complicated user behaviors cannot be simulated without real people, and experiment results are not always reproducible. Moreover, the scale may not be large enough for some applications. In addition, many PlanetLab machines may become unavailable at any time [12]. All these challenges imply that the flow study of management over the global Internet still has a long way to evolve.

However, within datacenters, the requirements of flow management come across some new issues. Performance requirements are much higher for the interconnection within datacenters. For example, aggregate traffic volume in datacenters of the High Energy and Nuclear Physics (HENP) community is expected to increase from 10 Gbps to the Tbps range [13]. However, since the computing and storage devices of a datacenter are physically co-located and their interconnect networks are usually centrally managed; its system model is much more controllable than the Internet. Aimed at bringing in a management layer upon current networking devices, several works have been proposed [14], [15]. Although these research contributed to drive the flow management more flexible and easier to deploy, smart dynamic mechanisms of flow management are still in desperate need of an appropriate theory and model.

In this paper, we exploit the possibility to address a twofold problem: performance improvement with reliability provisioning within datacenters. Based on a control theory analysis, we propose a Dynamic Feedback Control (DFC) model for the flow management problem within datacenters. The model is evaluated on the NS-2 [16] simulation platform. The experimental results illustrated a promising demonstration in the direction of designing new dynamic mechanism towards smart datacenters.

The remainder of this paper is organized as follows. Section

II summarizes the related works. Section III details the feedback control theory and Section IV describes the DFC model design. Section V analyzes the experimental results. In the last section, we draw a conclusion of our work.

II. Related Work

In this section, we summarized previous works on studying flow control problem with the feedback control scheme. The Feedback Control Theory is already utilized in the performance assurance mechanisms such as guarantees on bandwidth, latency or availability, all of which are considers as the requirement of QoS.

[17] presented a control-theoretic approach to reactive flow control. With deterministic and stochastic models, Packet-Pair rate probing technique and provably stable rate-based flow control schemes were proposed.

The Feedback Control Theory has been used in the TCP congestion problems to guarantee a fair share of resources among multiple TCP flows [18], [19]. These works attempt to apply control theory into the end-to-end framework to a certain extend.

T. F. Abdelzaher et al. [20] designed a control theory based approach to guarantee the bandwidth and latencyp of web server. At about the same time, IBM Laboratories also implemented a feedback controller in the queue manages of its Lotus Email Server [21].

The feedback control mechanism is also employed in the management of distributed real-time systems [22]–[24] to manage system resources, such as to control CPU utilization or to tune memory usage. These successful results of applying control theory to the control of the practical distributed systems imply that Feedback Control Theory can play a fundamental role in the control of dynamic systems, which also suggests that Feedback Control Theory is one of the essential theories for designing smart flow management mechanisms within datacenters.

Application Services Providers (ASP) also successfully used Feedback Control Theory to design their control layer to guarantee QoS. Examples are SwiFT [25] and ControlWare [26]. These interesting work applied Feedback Control Theory to the middleware architecture [27], which proves a model for distributed system services.

All these works successfully adopted feedback control into QoS assurance mechanisms, which suggests a promising way to support QoS within datacenters.

Recent years, there are a number of works to design scalable network architectures of datacenters. [28], [29] suggest datacenters based on the classic fat tree topology. Though DCell [30] and BCube [31] have proposed two specialized kinds of topologies for datacenter, these architectures are also implicit hierarchical in topology. As such, we take the classic fat tree topology in this paper. We believe other hierarchical topologies will be compatible with our work. A simple example of datacenter of fat tree topology is shown in Fig.1. As we can see, one fat tree topology datacenter includes three layers: Core-Layer, Aggregation-Layer and Edge-Layer. Specifically,

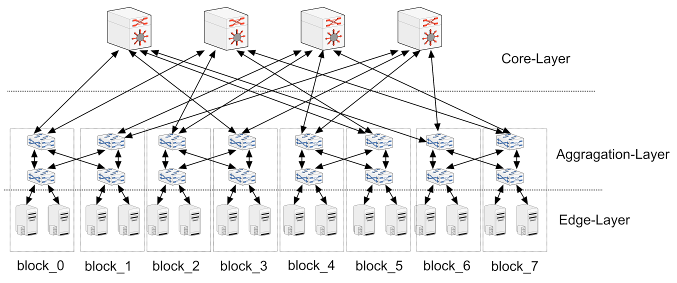

Fig. 1. A simple example for the fat tree topology

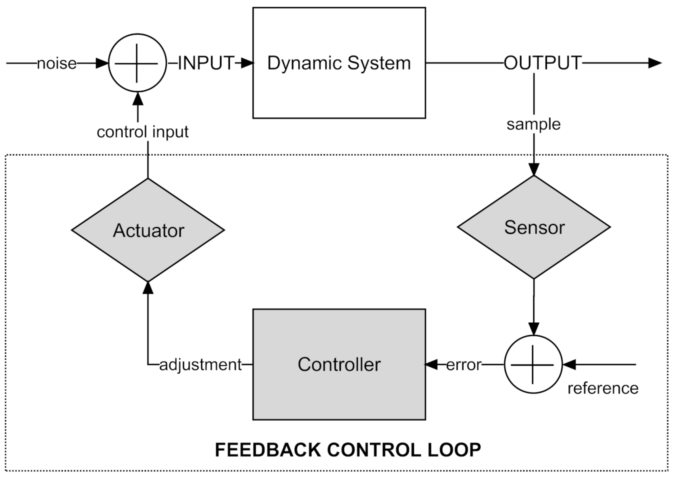

Fig. 2. Model of the Feedback Control System

at the Edge-Layer there are many groups of servers, such a group of servers is named as a block.

III. Feedback Control Mechanism

Feedback Control Theory [32] is quite adaptive to the behaviors of dynamic systems like datacenters. The term *feedback* means the controller affects the inputs variables by referring the output of the system, making a close-loop mechanism. Fig.2 shows a general model of *Feedback Control System*, which contains a *feedback control loop* and a *dynamic system*, or the controlled object. The feedback control loop, which is the main part of the Feedback Control System, consists of three components: *Sensor*, *Controller* and *Actuator*.

a. *Sensor*
Main task of the Sensor is to sample the output of the controlled system, and transfer the sampling results to the Controller. In datacenters, we use a sensor counter to watch the flow queue status of each server, where a busy state can be indicated.

b. *Controller*
Controller compares the output of Sensor and a reference input, to get the control error. With a predefined control algorithm, an adjustment value is calculated and sent to the Actuator.

c. *Actuator*
Actuator produces a signal to adjust the input of the controlled system based on the adaption results of Controller.

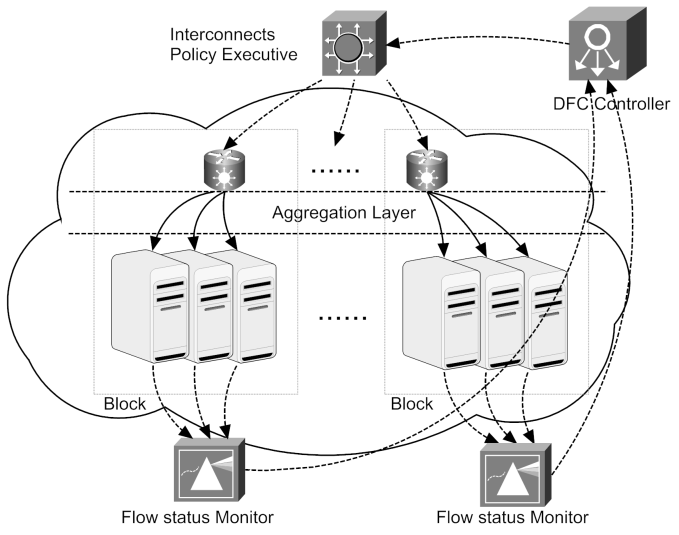

Fig. 3. Flow Management Model with Feedback Control

To manage the network flows within datacenters, a policy-aware switch should be used, such as [14].

With these three main components, Feedback Control Mechanism can be described generally as: *Controller* uses *Sensor* to watch the controlled system's output, and adjusts the input of the controlled system with the *Actuator*.

The transfer functions are described in Laplace transform [32]. If $G_1(s)$ represents the transfer function of the uncorrected open-loop system, $G_2(s)$ represents the transfer function of feedback loop. Then the global transfer function of close-loop system can be described in formula (1).

$$G(s) = \frac{y(s)}{v(s)} = \frac{G_1(s)}{1 + G_1(s)G_2(s)} \tag{1}$$

In (1), $v(s)$ and $y(s)$ represent the input and output of the control system separately.

From (1), the feedback loop is quite pivotal to the performance of the global control system, so the *Controller*'s adaption algorithm must be design appropriately.

IV. Dynamic Feedback Control

In this section we formulize the DFC model with Feedback Control theory, and to solve the problem of multiple-server controlling, we design an adaption algorithm.

A. Model Formulation

Instead of traditional high-performance computing center with expensive and special servers, today's Internet datacenters utilize commoditized computers and general switches to gain better price-performance trade-off [3]. Thus two critical requirements of the flow management within datacenters are performance and reliability. Novel management scheme can distribute the data flow dynamically to gain better resources utilization and improve the systemic performance. Reliability is also very important. Thus providing a stable service with unstable servers is also a main reason why DFC is proposed.

Though current datacenters may take quite different topologies, the two main components, servers and interconnects, always supply similar functions. Commercial servers process the data and store the resources, while interconnection devices like switches are responsible for arranging the path of the data flows. Since most datacenters are hierarchically constructed with the basic block unit, we give a feedback control model for datacenters, as shown in Fig.3. Each block utilizes a monitor to watch the status of servers and send the information to the DFC Controller. DFC Controller readjusts the workload using an adaption algorithm, based on the information collected. Besides, there are two channels in this model: Flow Channel in real lines and Control Channel in dashed lines.

- *Flow Channel*
  Flow Channel includes the interconnects and servers, it will support the data transmission path within datacenter. This channel is an original open-loop controlled system in this model.
- *Control Channel*
  Control Channel includes three components: the flow status monitor, the DFC controller and the interconnections policy executive. The monitor is deployed in the servers to gain enough information about the machines' status such as CPU status, bandwidth and memory utilization, etc. The executive controls the interconnection devices according to the flow management policies from the DFC controller.

We do not try to control the server status directly because almost all real network systems are dynamic and complicated. With controllable switches we can achieve a global optimization management by only controlling the network flows, which is quite economical and flexible in modern datacenters with few switches but lots of servers. With this condition, the feedback control loop can prove the datacenters to supply more stable services. Especially, when one server breaks down, DFC controller will migrate the unsettled workload to other available servers with the information from the monitor.

Suppose there is one server in the block, the input flow throughput is $p(t)$, the processing throughput is $q(t)$, then the queue length $l(t) = \int_0^t (p(\tau) - q(\tau))d\tau + C$ (C is constant), whose Laplace transform is $L(s) = \int_0^\infty l(t)e^{-st}dt, (s > 0)$. Suppose $p(t)$ and $q(t)$ are constant functions (for example, $p(t) = C_p, q(t) = C_q$), then $L(s) = \frac{C_p - C_q}{s^2} + \frac{C}{s}$. Since the Laplace transform of $p(t)$ is $P(s) = \frac{C_p}{s}$, we can get the block's transfer function, as described in(2).

$$G_1(s) = \frac{L(s)}{P(s)} = \frac{C_p - C_q}{C_p s} \tag{2}$$

With a linear feedback loop $G_2(s) = -\frac{C_f}{s}$, the global transfer function can be calculated as (3).

$$G(s) = \frac{G_1(s)}{1 + G_1(s)G_2(s)} = \frac{(C_p - C_q)s}{C_p s^2 - (C_p - C_q)C_f} \tag{3}$$

Here we get the transfer function for single-server block with linear feedback loop. To solve the problem of arranging workload in blocks with multiple servers, an adaption algorithm should be designed in the Controller.

B. Controller Design

The Controller decides the scheme to adjust the work flow load among different servers. To obtain an appropriate control algorithm, we base our design on the mathematic models of the problem. Assuming the datacenter can support a service with throughput $Q(t)$, and the i-th server can support a throughput as $X_i(t)$, thus $Q(t)$ and $X_i(t)$ will meet a equation in (4), which is also a boundary condition.

$$\sum_{i=1}^{n} X_i(t) = \lambda Q(t) \geq Q(t), \lambda \geq 1 \tag{4}$$

On the other hand, for a single server, too high work load will overwhelm the machine, while too light load will result in a waste of energy. With a dynamic work load given, the status of one server is a combination of CPU and memory utilization, and the usage rate of other resources. An optimal status point can be pre-configured for single server. However, the problem is more complicated with multiple servers together, because each server may own a special optimal status point. Thus the goal is to determine the control input $u(t)$ by solving (5).

$$F(\rho) = \min_{u(t)} \int_{0}^{\infty} \left((\rho - \overline{\rho})^T Q(\rho - \overline{\rho}) \right) dt, \rho = (\rho_1, \dots \rho_n) \tag{5}$$

Where $\overline{\rho} = \sum_{i=1}^{n} \frac{\rho_i}{n} I_n$, subject to

$$\sum_{i=1}^{n} \rho_i C_i(t) = Q(t), i = 1, \dots n \tag{6}$$

and

$$\rho_i(t) = \frac{X_i(t)}{C_i(t)}, i = 1, \dots n \tag{7}$$

In equation (7), $\rho_i(t)$ means the busy ratio of server i, while $C_i(t)$ represents the capacity of the i-th server at time t. Based on results in equation (4) to (7), an adaption algorithm is given in Algorithm 1 to distribute workload among multiple servers in the block.

This adaption algorithm can work right on condition that all servers are available. With disabled servers, a correctional scheme is adopted by the *Controller* to deal with these situations. If server i is failed, a special correction function should be introduced. One is described in (8).

$$\rho_i(t) = 1, 1 \leq i \leq n \tag{8}$$

Correction function (8) represents that server i is already with full load, thus no flow can be arranged to it.

C. Discussion

To build a general model, we try to simplify the architecture of datacenters first. For example, real datacenters may contains complicated structures between different levels of switches. In this paper we design the DFC model at the basic block unit, which can get more precise data. In other hand, there are some assumptions for the model itself, such as the *Sensors* are accuracy, and the *Actuators* are sensitive enough.

Algorithm 1 DFC Adaption Algorithm

```
Require: $\rho = (\rho_1 \dots \rho_n)$
 1: Init($\delta$) {Initialize an constant variable}
 2: Init($time_{sleep}$) {Initialize the adaption interval}
 3: Calculate $F(\rho)$ {see equation (5)}
 4: while $F(\rho) > \delta$ do
 5:     {Check $\rho, i = 1, \dots n$}
 6:     $\rho_{max} = GetMax(\rho_i)$
 7:     $\rho_{min} = GetMin(\rho_i)$
 8:     {Get id of the busiest and empties server}
 9:     $id_{max} = GetId(\rho_{max})$
10:     $id_{min} = GetId(\rho_{min})$
11:     {Adjust the workload}
12:     $X_{id_{max}} \leftarrow \frac{X_{id_{max}} + X_{id_{min}}}{2}$
13:     $X_{id_{min}} \leftarrow \frac{X_{id_{max}} + X_{id_{min}}}{2}$
14:     Sleep($time_{sleep}$)
15:     Calculate $F(\rho)$
16: end while
```

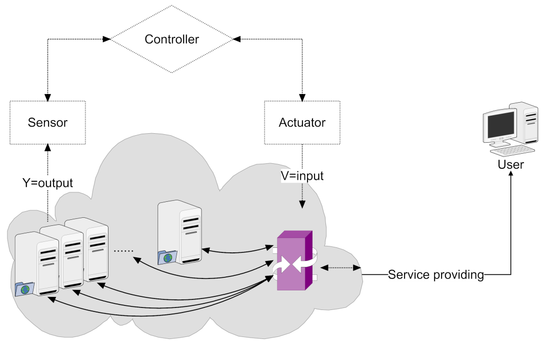

Fig. 4. A datacenter block with DFC flow management

V. Evaluation and Result

In this section, we use both the NS2 simulation platform and datacenter environment to evaluate the prototype of DFC. We focus on two aspects of the results: reliability and performance.

A. Simulation

To evaluate the reliability of our DFC flow management mechanism, we design a datacenter model with DFC based on the NS2 platform. Our DFC model is added into every block, which consists of 10 servers, as shown in Fig.4. The Controller can get the status information from Sensor every 0.5 s, and an adaption will be taken after that.

In our experiments, simulation parameters are chosen only for validation purpose. The throughput of each server is set to 100 Mbps, while the latency is 1 ms for all the links with a 1024 length DropTail queue, thus the maximal throughput for each block is 1000 Mbps. Each block is required to provide a total 500 Mbps TCP traffic service. The experiments test the datacenter's behavior under three modes. In the "normal" mode, all servers work normally during the simulation, "sick" mode means some servers will break down (In this simulation, 2 servers will break down at 1.5 s), while "DFC" mode means a DFC scheme will take effect in the datacenter to protect the throughput after some servers break down.

Simulation result is demonstrated in Fig.5. Triangle-curve represents the throughput under "normal" mode, diamond-curve represents the "sick" mode, while square-curve represents the "DFC" mode. Fig.5 compares the throughput under different modes. In "normal" mode, we can see a 500 Mbps throughput is provided by the test block from 0.5 s to 9.5 s. In "sick" mode, the throughput gets down to nearly 400 Mbps when 2 servers break down at 1.5 s. In "DFC" mode, the throughput curve also gets down at 1.5 s, however, due to the DFC flow management, the throughput then is improved near to the "normal" throughput of 500 Mbps at 2.0 s. This is the result of a redistribution of work flows after the DFC scheme takes action.

B. Experiment

To evaluate the efficiency of DFC mechanism, a prototype DFC datacenter is built. Topology of a block is similar as shown in Fig.4. Each block consists of 10 X86-64 commodity-class servers. For each server, the CPU is Intel Xeon L5335 (2.00 GHz, 4 cores on dual paths) with 4 MByte L2 cache and memory is of 8 GByte DDRII. Each server can support a throughput more than 500 Mbps, thus a more than 5 Gbps throughput can be provide with each block. In the experiment, to test the DFC model more accurately, we only let each block provide TCP traffic of 250 Mbps, which is far less than the maximal bandwidth it can support. The interval time between Controller's adaption also is set to 0.5 s, which is the same as in the simulation part.

The experiments are also carried out under three modes: "normal", "sick" and "DFC" mode. Experiment under each mode lasts for 30 s. In the "normal" mode, all servers work normally to test the normal running status of the block. For "sick" and "DFC" modes, one server will break down at 14 s. In particular, a DFC scheme is enabled at the "DFC" mode.

Fig.6 shows the throughput curves under three modes. The "normal" mode throughput holds about 250 Mbps for all 30 s, which indicates the capacity of the block. Throughput in "sick" mode remains nearly the full 250 Mbps from 0 to 14 s, but falls down when one server breaks down. In "DFC" mode, the throughput also falls from nearly 250 Mbps to about 215 Mbps at 14 s, however, as DFC scheme takes effect, the throughput recovers to nearly 250 Mbps about 2 seconds later.

In Fig.7, we also compare the throughput between datacenters using usual hash-based flow distribution mechanism and our DFC mechanism. The hash based mechanism (triangle block curve) attempts to distribute flows averagely using address hash, while DFC mechanism (diamond block curve) distributes the flows according to the status of servers. Result in Fig.7 indicates that DFC can improve the maximal processing capacity of datacenter when breakdown occurs. This is because DFC taps each server's processing potential while protecting it from overload.

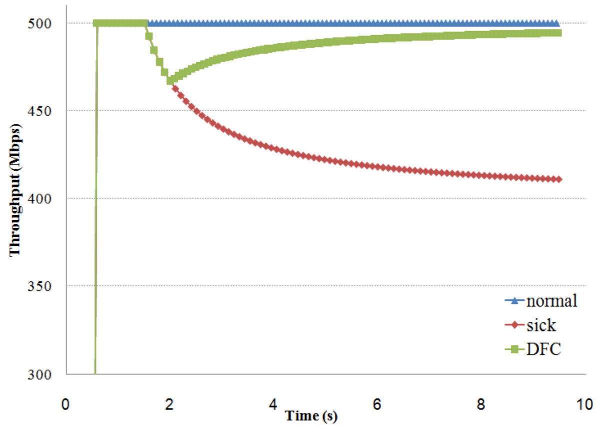

Fig. 5. Simulation result of reliability comparison

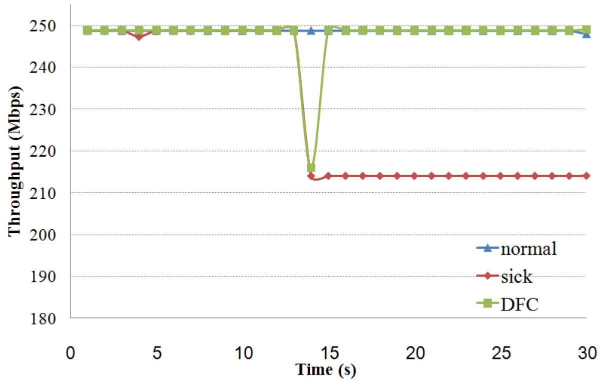

Fig. 6. Experiment result of reliability comparison

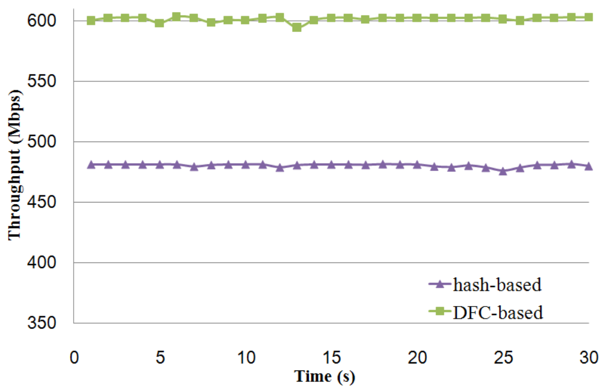

Fig. 7. Experiment result of performance comparison

C. Discussion

All the experimental results under three modes on both NS2 and phototype datacenter prove that DFC model provides a guarantee of the reliability, while at the same time, DFC improves the performance of datacenters. Though DFC scheme works well in our experiments, there is still something worthy investigation from the results. For example, the response sensitivity is ignored by the simulation for the platform limitation. This is important for applications like video meeting and on-line games, etc. Although TCP traffic is popular in modern datacenters, DFC is not designed specially for certain type of traffic, which will need more experimental supports.

VI. Conclusion and Future work

This paper proposes a Dynamic Feedback Control (DFC) flow management model within datacenters. The model uses a feedback control loop which consists of a *Sensor*, a *Controller* and an *Actuator* to manage the flows within datacenters. In order to improve the performance and reliability of the controlled datacenter, the adaption algorithms among multiple servers are also discussed. Simulation and real experimental results prove that the model can effectively improve the performance and guarantee the reliability of datacenters, especially when some servers break down. These results indicate a promising direction in optimization of flow management within datacenters, which is also a key contribution of our work. Furthermore, the DFC model works with general network flows and has a simple implementation. These features give the model the flexibility to be deployed in real and large-scale distributed network systems.

Though the experimental results are encouraging, there is still some interesting work to do. For example, the response sensitivity is ignored by the simulation for the platform limitation. In datacenters, this is important for applications like video meeting and on-line games, etc. The behavior analysis of the control loop under different types of real traffic is also an interesting research topic. The future work will include the research on sensitivity analysis for the DFC model and experiments in different types of real heterogeneous datacenters.

Acknowledgment

This work is funded by Tsinghua National Laboratory for Information Science and Technology (TNList) Cross-discipline Foundation (Traffic Management Technology Research for Datacenters), and the authors would like to thank all reviewers for their insightful suggestions and comments.

References

[1] IBM. (2008) Blue cloud project.

[2] Amazon. Amazon Elastic Compute Cloud (EC2). [Online]. Available: http://aws.amazon.com/ec2/

[3] L. Barroso, J. Dean, , and U. Hoelzle, "Web search for a planet: The Google cluster architecture," *IEEE micro*, vol. 23, no. 2, pp. 22–28, 2003.

[4] T. Benson, A. Anand, A. Akella, and M. Zhang, "Understanding data center traffic characteristics," in *Proceedings of the 1st ACM workshop on Research on enterprise networking*. ACM, 2009, pp. 65–72.

[5] R. Braden, D. Clark, and S. Shenker, "Integrated Services in the Internet Architecture: an Overview," IETF RFC 1633, June 1994.

[6] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An Architecture for Differentiated Services," IETF RFC 2475, December 1998.

[7] D. Grossman, "New Terminology and Clarifications for Diffserv," IETF RFC 3260, April 2002.

[8] D. D. Clark, "The design philosophy of the DARPA Internet protocols," *Symposium proceedings on Communications architectures and protocols*, p. 114, 1988.

[9] PlanetLab. [Online]. Available: http://www.planet-lab.org/

[10] L. Peterson, T. Anderson, D. Culler, and T. Roscoe, "A blueprint for introducing disruptive technology into the Internet," *ACM SIGCOMM Computer Communication Review*, vol. 33, no. 1, pp. 59–64, 2003.

[11] *Operating system support for planetary-scale network services*. USENIX Association, 2004.

[12] N. Spring, L. Peterson, A. Bavier, and V. Pai, "Using PlanetLab for network research: myths, realities, and best practices," *ACM SIGOPS Operating Systems Review*, vol. 40, no. 1, p. 24, 2006.

[13] F. Berman, G. Fox, T. Hey, J. J. Bunn, and H. B. Newman, "Data intensive grids for high energy physics," *Grid Computing: Making the Global Infrastructure a Reality*, pp. 859–906, 2003.

[14] Dilip A. Joseph, Arsalam Tavakoli, and Ion Stoica, "A Policy-aware Switching Layer for Data Centers," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 4, pp. 51–62, 2008.

[15] N. Mckeown, S. Shenker, T. Anderson, L. Peterson, J. Turner, H. Balakrishnan, and J. Rexford, "OpenFlow: Enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008.

[16] NS2. [Online]. Available: http://nsnam.isi.edu/nsnam/index.php/Main_Page

[17] S. Keshav, "A control-theoretic approach to flow control," *ACM SIGCOMM Computer Communication Review*, vol. 25, no. 1, p. 201, 1995.

[18] C. V. Hollot, V. Misra, D. Towsley, and W. Gong, "A control theoretic analysis of RED," in *IEEE INFOCOM*, vol. 3. Citeseer, 2001, pp. 1510–1519.

[19] Yu Lin, Haitao Wu, Rui Fan, and Shiduan Cheng, "Modeling multiple TCP connections established between a busy server and many receivers," in *Global Telecommunications Conference, 2003. GLOBECOM '03. IEEE*, January 2003.

[20] Abdelzaher, T.F., Shin, K.G., Bhatti, and N., "Performance guarantees for web server end-systems: A control-theoretical approach," *IEEE Transactions on Parallel and Distributed Systems*, pp. 80–96, 2002.

[21] S. Parekh, J. Hellerstein, T. S. Jayram, N. Gandhi, D. Tilbury, and J. Bigus, "Using control theory to achieve service level objectives in performance management," *Real-Time Systems*, vol. 23, no. 1, pp. 127–141, 2002.

[22] C. Lu, X. Wang, and X. Koutsoukos, "Feedback utilization control in distributed real-time systems with end-to-end tasks," vol. 16, no. 6. Citeseer, 2005, pp. 550–561.

[23] J. Stankovic, T. He, T. Abdelzaher, M. Marley, G. Tao, S. Son, and C. Lu;, "Feedback control scheduling in distributed real-time systems," in *Proceedings of the 22nd IEEE Real-Time Systems Symposium (RTSS'01)*. Citeseer, 2001, p. 59.

[24] A. Storm, C. Garcia Arellano, S. Lightstone, Y. Diao, and M. Surendra, "Adaptive self-tuning memory in DB2," in *Proceedings of the 32nd international conference on Very large data bases*. VLDB Endowment, 2006, p. 1092.

[25] A. Goel, David, A. Goel, D. Steere, D. Steere, C. Pu, C. Pu, J. Walpole, and J. Walpole, "SWiFT: A feedback control and dynamic reconfiguration toolkit," *Oregon Graduate Institute of Science & Technology, Beaverton, OR*, 1998.

[26] R. Zhang, C. Lu, T. Abdelzaher, and J. A. Stankovic, "Controlware: A middleware architecture for feedback control of software performance," in *International Conference on Distributed Computing Systems*, vol. 22. IEEE Computer Society; 1999, 2002, pp. 301–310.

[27] P. A. Bernstein, "Middleware: a model for distributed system services," 1996.

[28] A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz, and P. Pat, "VL2: A scalable and flexible data center network," *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 4, pp. 51–62, 2009.

[29] R. Niranjan Mysore, A. Pamboris, N. Farrington, N. Huang, P. Miri, S. Radhakrishnan, V. Subramanya, and A. Vahdat, "PortLand: a scalable fault-tolerant layer 2 data center network fabric," *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 4, pp. 39–50, 2009.

[30] C. Guo, H. Wu, K. Tan, L. Shi, Y. Zhang, and S. Lu, "DCell: A scalable and fault-tolerant network structure for data centers," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 4, pp. 75–86, 2008.

[31] C. Guo, G. Lu, D. Li, H. Wu, X. Zhang, Y. Shi, C. Tian, Y. Zhang, and S. Lu, "Bcube: A high performance, server-centric network architecture for modular data centers," vol. 39, no. 4. ACM, 2009, pp. 63–74.

[32] J. Doyle, B. Francis, and A. Tannenbaum, *Feedback control theory*. Citeseer, 1992.