International Journal of Innovative Computing, Information and Control
Volume **8**, Number **3(B)**, March **2012**

ICIC International ©2012 ISSN 1349-4198
pp. **2015–2032**

IMPROVING ROUTING STATE CONSISTENCY AND REDUCING OVERHEAD FOR MIGRATION-BASED LOAD BALANCE SYSTEMS IN STRUCTURED OVERLAY NETWORKS

Chyouhwa Chen, Shi-Jinn Horng* and Kevin Chang

Department of Computer Science and Information Engineering
National Taiwan University of Science and Technology
No. 43, Sec. 4, Keelung Rd., Taipei 106, Taiwan
*Corresponding author: horngsj@yahoo.com.tw

Received August 2010; revised December 2010

Abstract. *In DHT-based peer to peer systems, migration-based load balancing schemes employing virtual servers as their fundamental mechanism have been shown to be the most viable strategy for load balancing when workload may shift dynamically as time progresses. However, all previous proposals must employ a large number of virtual servers to be effective, resulting in two severe problems. First, they incur excessive overheads. Secondly, they induce significant inconsistency in DHT routing state due to node churn. To solve the problems, we propose the inclusion of virtual server management strategies and an active stabilization mechanism. The virtual server management strategies intelligently manage the discard and creation of virtual serves. In our more aggressive virtual server discard strategy, the overheads of virtual server-based systems can be made to approach that of non-virtual-server-based systems, with a slight increase in workload imbalance. In our adaptive virtual server management strategy, it can maintain a level of virtual servers depending on system utilization, and system workload can be made to be as balanced as that in previous proposals that employ more virtual servers. Coupled with the active stabilization mechanism, our proposal solves the routing state inconsistency problem, while reducing the operating overheads significantly. All of our proposals can improve query success performance, as we always employ less virtual servers.*

Keywords: DHT-based peer to peer systems, Virtual servers, Active stabilization mechanism, Workload imbalance, System utilization, System workload

1. Introduction.

Structured overlay networks based on distributed hash tables (DHT) have been found to be a versatile paradigm and have been widely applied in many domains [7,9,13,16-18,20,23,30]. These systems are characterized by autonomy, scalability, among many other desirable features. However, scalability of the systems critically depends on the extent to which the systems can effectively balance the workload among the peers so that the collective capacity of all peers can be utilized. One of the fundamental objectives for workload balance is peer network bandwidth utilization [1,5,8,12,15,19,21,22]. The other popular objective is the storage space when objects are to be stored [7,15,19,20]. In this paper, we consider the load balance problem when the objective is network bandwidth. The DHT ID-space, or DHT space for short, model for structured overlay networks considered in this paper is the popular logical ring model, which has been used in Chord, among others [18]. In this model, the peers and objects are placed on the ring and peers are responsible for objects whose IDs fall in non-overlapping parts of the ring, also called zones or segments [21]. The load balance problem in the ring model has typically been viewed as a zone partitioning problem; that is, the goal is to partition the zones among the nodes so that the DHT space is evenly partitioned, or proportional to each node's capacities. Following [21], let $f_{\max}$ be the ratio of the largest zone size to the average zone size. It is well known that $f_{\max}$ is $\Theta(\log n)$ with high probability if a random location is selected as a peer's ID [11]. A natural extension is to sample d random points in the ID space and choose the largest zone to split into two halves. A different strategy is to enable each peer to instantiate multiple logical instances, called virtual servers (VS), and let each VS be responsible for a zone. It is known that, in the case of homogeneous system capacities, having $\Theta(\log n)$ instances per physical node reduces $f_{\max}$ to a constant factor [7].

Note that the zone partitioning approach to the load balance problem [21] can only deal with applications whose workloads are static; that is, the system workload does not shift in the DHT ID space with time. As the workload of real overlay network applications may evolve over time, a balanced overlay network may become imbalanced as the system evolves. To solve this issue, systems based on VS migrations must be used, including *k-choices* [12], the Many-to-Many algorithm with dislodge (abbreviated as M2M) [15,19], Hsiao's proposal [8] and GBS [22], because they can dynamically move the VS to different physical nodes if necessary. Experiments have shown these systems to be effective for load balancing in networks whose workload may evolve dynamically [1,5,8,12,15,19,22]. In the following, Hsiao's proposal [8] and GBS [22] will not be discussed further as they assume a fixed set of virtual servers, and do not include virtual server management actions.

Even though these systems have been proven to enable high load balance performance, they all require a large number of virtual servers to effectively maintain the balance of workloads. As an example, we performed a detailed study of the changes in the number of VSs in M2M and *k-choices* in scenarios when system total workload is either light or heavy. Figure 1 depicts the evolution of average number of virtual servers per peer in M2M and *k-choices* (denoted by KC in the figure) as time progresses. M2M employs a fixed number of VSs, while *k-choices* tries to be adaptive. The behavior of *k-choices* is particularly interesting. In Figure 1(a), when the workload is light, the number of VSs increases continually over time in *k-choices* as time progresses, while in Figure 1(b), the number of VSs remains relatively unchanged, yet is significantly fewer than that in M2M.

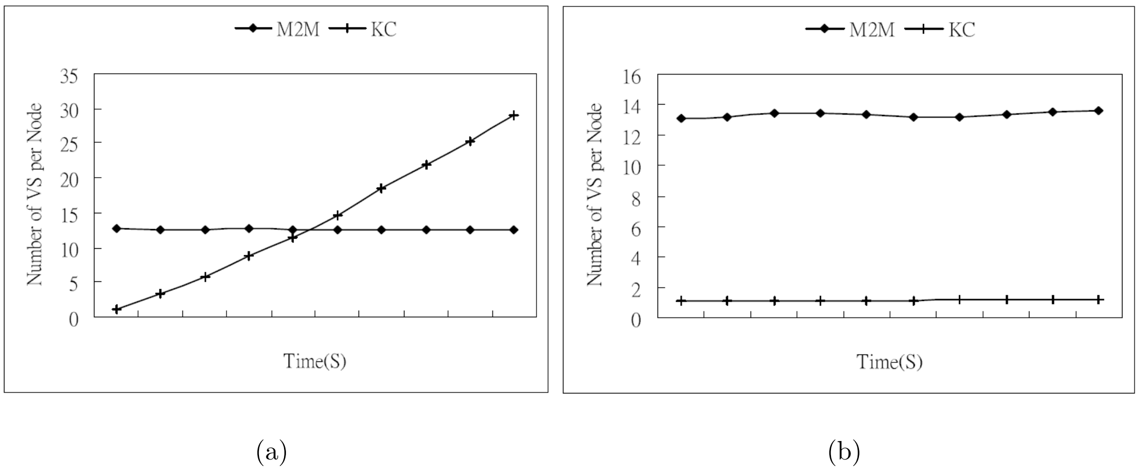

FIGURE 1. (a) Total number of virtual servers when workload is light; (b) total number of virtual servers when workload is heavy

The fact that all previous migration-based systems must employ a large number of VSs to achieve their load balance objective creates two major problems. The first major problem is that the large number of VSs incurs significantly more overhead than non-VS-based systems. For each virtual server, storage space for a routing table must be maintained. More importantly, the average route length between peers increases when a larger number of VSs are involved in the system.

The second major problem in migration-based systems is the excessive inconsistency of the routing state. Even though the Chord stabilization procedures *fix_finger*() and *stabilize*() have been shown to reasonably maintain the consistency of finger tables [13,14,18], in a migration-based system, periodic VS migration induces additional, and much more severe routing state inconsistency problems. The problem is worse when the system utilization is high or attacked [26-29]. In Figure 2, we depict the average percentage of incorrect finger table entries versus query request rate of M2M. As many as 30% of the finger table entries are incorrect on average under the original stabilization procedures. The routing state inconsistency problem is severe, and requires the incorporation of more mechanisms than the standard Chord stabilization procedures to combat the problem.

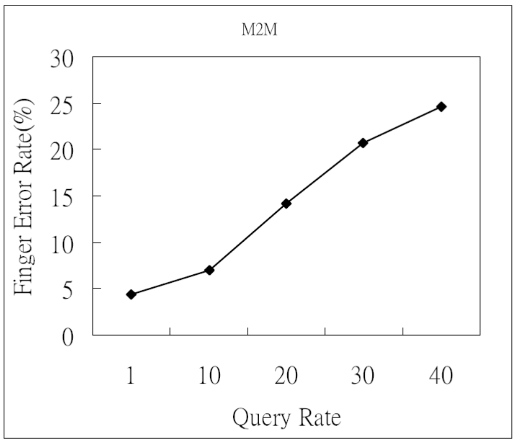

FIGURE 2. Percentage of finger table entries affected by each round of the load balancing actions

The unique features of our work are two-fold. Our primary contribution is that we propose a number of VSM strategies that are of fundamental importance to migration-based load balance strategies. The strategies include a number of pure discard VSM strategies and an adaptive VSM strategy. Our secondary contribution is that we demonstrate the necessity of the inclusion of the reverse finger table (RFT) mechanism [9,13] in a VS-based load balancing system to maintain the routing state consistency when churn rate is high. The RFT mechanism is shown to be essential for the migration-based systems.

The organization of this paper is as follows: in Section 2, related works in the literature are surveyed; in Section 3, we present the design and justification of the proposed VSM strategies in detail; in Section 4, the performance of the related systems is compared through an extensive set of simulation experiments to demonstrate the superiority of our proposals; in Section 5, we present the conclusions of this work and possible future directions.

2. **Related Works.**

For systems employing virtual servers for load balancing, the dimensions of design include join-time VS placement or run-time VS migration placement, semi-distributed operation via directory node or completely distributed operation, and virtual-server management strategies. In the following, we examine previous works that are related to ours.

k-choices, along with other static load balancing schemes [21] relies mainly on intelligent placement of VSs as nodes join the P2P network. On the other hand, M2M-like systems rely on periodic VS migration to achieve load balance. *k-choices* and Y0 [1] are designed to handle the heterogeneous node capacity aspect of load balance problem, through intelligent ID selection for VSs at node join time. *k-choices* creates k VSs at node join time. For each VS, k random locations are probed to determine a best one at which to join. Y0 also instantiates $\Theta(c_v \log N)$ VSs for each node of capacity c_v. Our focus is on M2M-like systems, which also instantiates a number of VSs proportional to the capacity of a node. When a node n with capacity c_n joins the system, the number of VS m_n to instantiate is $m \times (c_n/c)$, with m being a system parameter, and c being the average node capacity. In contrast to *k-choices*, M2M places the VSs at random locations, and rely on periodic VS reassignment to migrate VSs from overloaded nodes to under-loaded nodes. In this manner, the system can reach a load balanced state.

Another main design dimension is the decision to incorporate directory nodes or not, which keeps track of load imbalance information of peers and make load balance decisions. GBS and M2M include a directory component in their protocols. In M2M, one or more of the VSs may be removed from an overloaded physical node and moved to other physical nodes. In effect, the involved address ranges are transferred to other physical nodes, with the IDs of the moved VSs unchanged. On the other hand, *k-choices*, Hsiao's proposal [8] and Y0 [1] do not. The directory nodes are a logical component that serves as rendezvous points for peers to exchange load imbalance information. Strategies that employ directories have higher load balance performance. Of course, maintaining the directory components require an extra set of protocols and overhead for their creation and destruction.

A very important design dimension not explicitly discussed in most of the previous works is the VS management strategy. A VS management strategy comprises the split and merge actions taken to control the number of VSs employed in the system. Both *k-choices* and M2M employ VSM actions, while Y0 and GBS do not. As our work is based on the M2M framework, we describe the M2M scheme in more details. M2M's action tries to keep the total number of VSs relatively constant, while in keeping with its main load balancing purpose. First, when the average number of VS in a directory is smaller than a lower threshold (< 0.75m), the largest VS is split in half. The assignment of the new VS is determined by the VS reassignment algorithm. Secondly, when the average number of VSs per node in a directory is larger than an upper threshold (> 1.25m), a least-loaded VS is removed by the directory node. The general conclusion drawn by both M2M and *k-choices* is that employing more VSs improves the load balance performance of the systems, with decreasing marginal benefit as the number of VSs increases. As the goal of these systems is load balance, both of the systems tend to create a large number of VSs, incurring excessive routing overheads, which is the first major problem suffered by current migration-based systems.

Finally, in [6], the authors investigate how a secure coprocessor may be employed for voting-based applications. In [10], the authors proposed an authentication protocol for use in a multiple server distributed setting. Another authentication protocol for use in a multiple server setting under a time-bound is proposed in [4]. These works examine security issues that are not addressed in this paper, but are potentially useful for future enhancement of our results.

3. Proposed Novel Virtual Server Management Policies and Active Stabilization.

In this section, we first propose a number of *virtual server management* (VSM) policies that are designed to control the number of VSs used in the system without inducing much adverse impact on system performance. Then, we argue for the inclusion of an *active stabilization* (AS) component to address the routing state inconsistency problem. The AS component restores an inconsistent routing state to a consistent state in a speedy fashion.

We will present our proposals in terms of the M2M framework [15,19]. However, the concept of VSM is equally important to *k-choices* and other migration-based systems. The pseudo codes executed at physical nodes and directory nodes in M2M are shown in Figures 3 and 4, respectively. With respect to the code for a physical node, shown in Figure 3, it is the same as that in M2M, except for the inclusion of a new physical node level VSM component (`Node-VSM()`). At node join time, a node instantiates $m = \Theta(c_v \log N)$ VSs in proportional to its capacity c_v at random locations. It then selects a directory node to register by invoking `RandomDirectory()`, which probes two random directory nodes and returns one with less total number of registered nodes. It then registers with one by sending information regarding its own capacity and VSs instantiated. Periodically, a physical node receives VS reassignments directives from a directory and transfer the VS `v` to another peer n' via `PerformTransfer()`. The `Node-VSM` component will be discussed later.

```
Algorithm for Physical Node
Initialization:
1)  Instantiate $m = \Theta(c_v \log N)$ VSs at random locations
2)  Send ($c_v$, $\{vs_1, ..., vs_m\}$) to RandomDirectory()
Periodically: Every time period $T$
1)  Receive a list of transfer directives from a directory node
2)  PerformTransfer(v, $n'$) for each transfer directive v $\rightarrow n'$
3)  Report ($c_v$, $\{vs_1, ..., vs_m\}$) to RandomDirectory()
4)  Node-VSM();
```

FIGURE 3. Algorithm executed by physical nodes

The code executed by a directory node is depicted in Figure 4. The most important function of a directory node is to determine a new VS reassignment, the associations between VSs and the physical nodes, so that system workload imbalance is reduced, embodied in the `ReassignVS()` function. The specific VS reassignment algorithm is not relevant in this paper. In M2M, it suffices to say that `ReassignVS()` greedily moves heaviest VSs from overloaded physical nodes to under-loaded physical nodes. Details of the algorithm can be found in [15,19]. Due to the importance of this procedure, we note that a number of other VS reassignment algorithms have been proposed in the literature [5].

After a new reassignment is found, directory nodes informs the physical nodes of the new assignment via transfer directives, and the physical nodes performs the actual actions of migrating affected VSs from one physical node to another. In this paper, two components are added for directory nodes – the `Directory-VSM()` component and the `ActiveStabilize()` component. In the following, we discuss the details in their design.

We first describe below a number of concrete VS pure discard algorithms. We then describe two directory level policies that combine VS discard and creation mechanisms. The VS pure discard algorithms are the simplest to implement, while achieving excellent query success performance and suffering from load balance degradation only slightly. Specifically, we investigate the following three algorithms in this paper.

```
Algorithm for Directory
Initialization:
1)  I ← { }
Periodically: Every time period $T$
Input: time period $k_p$
1)  Directory-VSM()
2)  reassignment ← ReassignVS (I, $k_p$)
3)  ActiveStabilize()
4)  Schedule transfers according to reassignment
ReassignVS(I, $k_p$){
//Compute a reassignment of VSs for physical nodes in the current directory so
that the workloads on the physical nodes are as balanced as possible
}
```

Figure 4. Algorithm executed by directory nodes

√ `Node-VSM-ED(Physical Node-level Eager Discard)`: This is an algorithm implemented at the physical node level, in the `Node-VSM()` module. The pseudo code is shown in Figure 5. Each physical node discards one VS periodically without regard to its consequence on the workload of the successor VS, in a completely distributed manner. The VS being discarded is the lightest one that is associated with it.

√ `Node-VSM-PD(Physical Node-level Prudent Discard)`: This is an algorithm implemented at the physical node level, in the `Node-VSM()` module. The pseudo code is shown in Figure 6. This algorithm is similar to the previous one. A physical node discards also periodically, in a distributed manner, one lightest VS. However, the VS being discarded is the lightest among those whose deletion do not overwhelm their successor VSs.

√ `Dir-VSM-CSD (Directory-level Capacity Sensitive Discard)`: This policy is implemented on a directory node, in the `Directory-VSM()` module. The pseudo code is shown in Figure 7. Each directory aggressively discards VSs on small capacity nodes. It discards VSs less aggressively on large capacity ones, only when their utilization is larger than the directory node average.

Note that the `Node-VSM-`* algorithms are invoked by the `Node-VSM()` procedure in Figure 3, while the `Dir-VSM-`* algorithms are invoked by the `Directory-VSM()` procedure in Figure 4.

```
Algorithm:  Node-VSM-ED()
//Completely Distributed Eager discard
Execution entity: individual physical nodes
Procedure:
1)  Sort the workload of all VSs on a physical node;
2)  Discard $\alpha$ lightest VSs on itself if the number of VSs is > VSmin
```

Figure 5. Node-level eager discard algorithm

We next present two directory level policies that combine VS discard and creation mechanisms. Even though we have also investigated a completely distributed node-level VS Discard and Creation policy based on mechanisms inspired by *k-choices*, we do not present its results here as its performance is inferior to those presented. The first one is similar to the original M2M algorithm.

```
Algorithm:  Node-VSM-PD()
1)  Discard $\alpha$ lightest VSs on itself only if the successor VS
can accept the workload and if the number of VSs is $>$ VSmin
```

FIGURE 6. Node-level prudent discard algorithm

```
Algorithm:  Dir-VSM-CSD()
//Capacity Sensitive discard
1)  For all physical nodes $n$ in this directory
2)      if ($n$'s utilization is $>$ directory average utilization $\|$ $n$'s capacity
is $<$ directory average capacity)
3)          Discard $\alpha$ lightest VSs on $n$ if the number of VSs on $n$ is $>$ VSmin
```

FIGURE 7. Directory-level capacity-sensitive discard algorithm

```
Algorithm:  Dir-VSM-M2M()
1)  while average number of VSs per node is $>$ 1.25m
2)      Remove the least loaded VS
3)  while average number of VSs per node is $<$ 0.75m,
4)      Split the largest loaded VS in half
```

FIGURE 8. Directory-level M2M VSM algorithm

```
Algorithm:  Dir-VSM-Adaptive(){
//Try to create VS if necessary
1)  If the average utilization of physical nodes in the directory is $\gamma\%$
2)      Split $\beta$ heaviest VS into half
//Try to discard unnecessary VSs using the Node-VSM-PD – strategy
3)  for all physical nodes in this directory
4)      Discard $\alpha$ lightest VSs on a physical node only if the successor VS
can accept the workload and if the number of VSs is $>$ $\text{VS}_{\text{min}}$
```

FIGURE 9. Adaptive directory-level combined creation and discard algorithm

$\surd$ `Dir-VSM-M2M (Directory-level M2M VSM algorithm)`: This algorithm is similar to the original M2M VSM algorithm executed by directory nodes, except that in our algorithm the VSM actions are taken before VS reassignment. The pseudo code is shown in Figure 8. Including an algorithm similar to the original M2M VSM algorithm is mostly for completeness' sake in performance comparisons.

$\surd$ `Dir-VSM-Adaptive (Directory-level Adaptive VSM algorithm)`: The pseudo code of Dir-VSM-Adaptive algorithm is shown in Figure 9. In the algorithm, directory nodes perform VSM actions periodically. The idea is that when the average utilization of physical nodes in a directory reaches a certain threshold $\gamma\%$, a number β of VSs are created by splitting the heaviest VSs in the directory. In addition, all possible excess VSs are discarded as usual using a strategy similar to the Node_VSM_PD algorithm. Both VS discard and creation are performed in each invocation of the algorithm, resulting in speeding corrective actions even when some locations are overloaded and simultaneously some other locations are under-loaded.

The Reverse Finger Table Active Stabilization Mechanism. To deal with the routing state consistency problem, we propose to augment each node n with a reverse finger table (RFT), which is also called inverse finger table in [14] and has been studied in detail recently [9]. A RFT contains the set of nodes that use n as a neighbor. Each node stores information regarding these nodes in its RFT, along with their associated physical node addresses. Maintenance of RFT is actively performed. It is implemented via the call to `ActiveStabilize` () in the algorithm for directory nodes, as depicted in Figure 4. The `ActiveStabilize` algorithm involves simply a call to *RFT_manage*() procedure, which performs two important operations. First, after a decision is made to reassign a VS n from an old physical node to a new one, both the old node and the new node are informed of the reassignment. Each node then sends an update packet to all physical nodes referenced to by the RFTs in the VSs associated with this physical node. All affected VSs can then update their finger tables immediately, preserving the routing state consistency. Secondly, when a VS is discarded, its associated physical node is informed, which then performs a similar action to update the finger tables of affected VSs.

```
Algorithm: ActiveStabilize()
1)  RFT_manage()
```

Discussion of Our Strategies. A number of issues are not addressed in this paper. First, security issues are not addressed in this paper. Security related issues must be addressed before our design can be deployed in the real world. A number of research works [4,6,10] have proposed schemes that may be fruitfully incorporated into our design and will be explored in the future. Secondly, in terms of computation cost, our strategies add only overheads for reverse finger table maintenance. However, as will be shown in the performance evaluation section, the cost is minimal.

4. **Performance Evaluation.**

We compare the performance of the proposed VSM algorithms against the original M2M and *k-choices* schemes to illustrate the benefits of our schemes in this section. The evaluations are done using the P2Psim simulator [25], for the case when the target for load balance is network bandwidth. In the experiments, for M2M and *k-choices*, the active stabilization operations are not performed, so that their results are consistent with those reported in the literature.

4.1. **Experiment setup.**

To obtain a complete view of the performance, we have used metrics for evaluating load balance, query success and overhead performance of the algorithms, including the following. 1) Routing table consistency, which is defined as the average percentage of erroneous finger table entries. 2) 99th percentile node utilization, which is a measure for the load balance performance. The maximum of 99th percentile utilization is defined to be the maximum of the 99th percentile utilization among all physical nodes in every round over a period of time. 3) Query success ratio, which is defined as the percentage of successfully routed queries among all queries issued. 4) Query route length: The average number of hops traversed over the P2P network to reach its desired destination for an issued query. 5) Stabilization overhead: The total number of packets for routing table stabilization, measured by counting messages sent in *fix_finger*(), *stabilize*() and *RFT_manage*().

The experiments are divided into three sets. In the first set, we show that the simple pure discard VSM algorithms are effective in most situations. The pure discard VSM algorithms are important because they can be efficiently implemented in a distributed manner. In the second set, we show that the adaptive algorithm with combined discard and creation operations is preferable in situations where the most load balanced system state is desired. However, the adaptive algorithm can only reap their benefits when they are implemented on the directory nodes. Finally, in the third set of experiments, we demonstrate the important role of the active stabilization mechanism when VSs are constantly being migrated and created/discarded. In the following, we give more details on the three sets of experiments.

To explore the performance of the various strategies when faced with heterogeneity in the workloads, lookup query destinations are generated using Zipf's distribution with varying parameters. The queries are issued by physical nodes every second, with rates ranging between 10 $\sim$ 40 queries/sec. To create node capacity heterogeneity, node capacities are modeled by the Pareto distribution [24], which is characterized by a shape α and a scale β parameter. Infinite variance occurs when the shape parameter value is 2, and that the variance decreases as α value increases. To generate distributions with variance from ∞ to 0, α is varied between 2 and ∞. To maintain a fixed mean μ, β values are then calculated according to Pareto distribution's formula $\mu = \alpha\beta/(\alpha - 1)$. The peer churn process is also generated using Pareto-distributed lifetime process, patterned after [12]. Table 1 lists the parameter values used. The third column lists the default values for each parameter if not stated.

TABLE 1. Parameter values used

Parameters	Range	Default
Number of nodes	$\sim$ 2048	$\sim$ 2048
Number of queries/second	1 $\sim$ 40	10
Node birth/death process	Pareto	Pareto
Average lifetime of physical nodes	15min $\sim$ 120min	60min
Initial # of VSs/node for all strategies	12	12
α_c: shape of node capacity	2.0 $\sim \infty$	2
β_c: scale of node capacity	200 $\sim \infty$	200
Lookup Query distribution	Zipf 0 $\sim$ 4.8	Zipf 1.2
period of VSM actions	60sec	60sec

4.2. **Experiment results.**

A. Performance of Pure Discard VSM Algorithms. We examine the performance of the simple pure discard VSM algorithms in this section.

A.1. Routing table consistency. Figure 10 depicts routing table consistency results in terms of the average percentage of erroneous finger table entries versus query rate for Dir-VSM-ED, Node-VSM-PD, Dir-VSM-CSD and Dir-VSM-M2M algorithms, when the mean node lifetime is 15 minutes.

First, observe that more finger table entries become erroneous for all three algorithms as query rate increases. This is not surprising because system stabilization consumes network bandwidth, too, and therefore, routing table consistency is adversely impacted by the amount of query traffic in the system. Secondly, observe that both Dir-VSM-ED and Node-VSM-PD algorithms keep the routing tables in a more consistent state than Dir-VSM-M2M for all query rates. This is because Dir-VSM-M2M employs many more VSs and does not employ the active stabilization mechanism. The improvement is especially significant at high query rates, when routing tables become highly inconsistent under Dir-VSM-M2M, while Dir-VSM-ED, Dir-VSM-CSD and Node-VSM-PD show much better resilience to the bandwidth strain under higher query workload.

The results for routing table consistency versus average node lifetime (in minutes) are shown in Figure 11, when query rate is fixed at 40. Not surprisingly, the higher the churn rate, the less consistent the routing tables are for all algorithms. For Dir-VSM-M2M, the original Chord stabilization procedures can cope with node churn reasonably well only

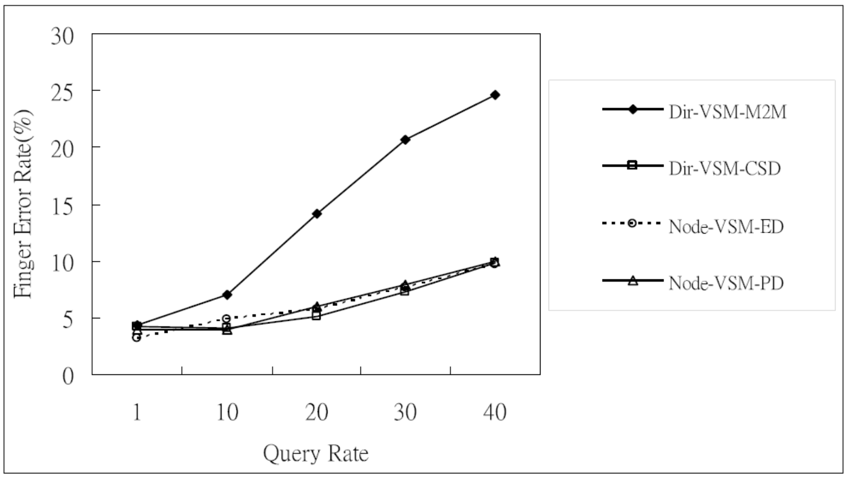

Figure 10. Percentage of incorrect finger table entries versus query rate

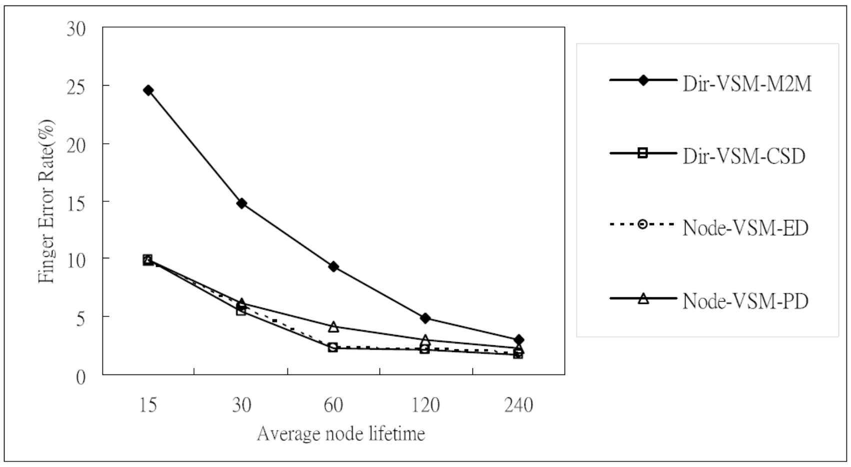

Figure 11. Percentage of incorrect finger table entries versus average node lifetime

when the average node lifetime is longer than 60 minutes. For Dir-VSM-ED, Dir-VSM-CSD and Node-VSM-PD, they maintain the consistency of the routing tables reasonably well across the spectrum of node life times.

A.2. Load balance performance under varying system load. With significantly fewer VSs in our system, the best we can hope for is for our proposals to achieve comparable load balancing performance. The load balance performance results for the related systems are illustrated in Figure 12, using the 99 percentile node utilization metric.

Clearly, Dir-VSM-M2M achieves the best load balance performance, while Dir-VSM-ED and *k-choices* perform worse at high query rates. Surprisingly, Node-VSM-PD has exceeded our expectations by achieving comparable load balance performance as Dir-VSM-M2M, which employs 12 VSs on average on each physical node, while, as will be shown later, Node-VSM-PD and Dir-VSM-ED employ close to 1 VS on average.

A.3. Average number of VS and query route length. The average number of VSs and average route length for query lookups results under the various VSM algorithms are shown in Figures 13 and 14, respectively. *Average number of virtual servers* is defined as the total number of remaining VSs divided by the total number of physical nodes. *Average route length* is defined as the average number of hops traversed for successful query lookups. These reflect the network efficiency of the systems. First, Dir-VSM-M2M

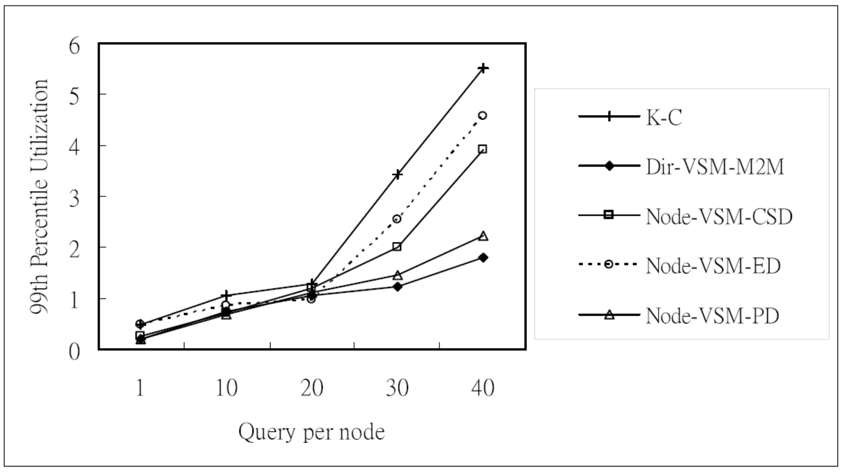

Figure 12. The 99 percentile node utilization performance

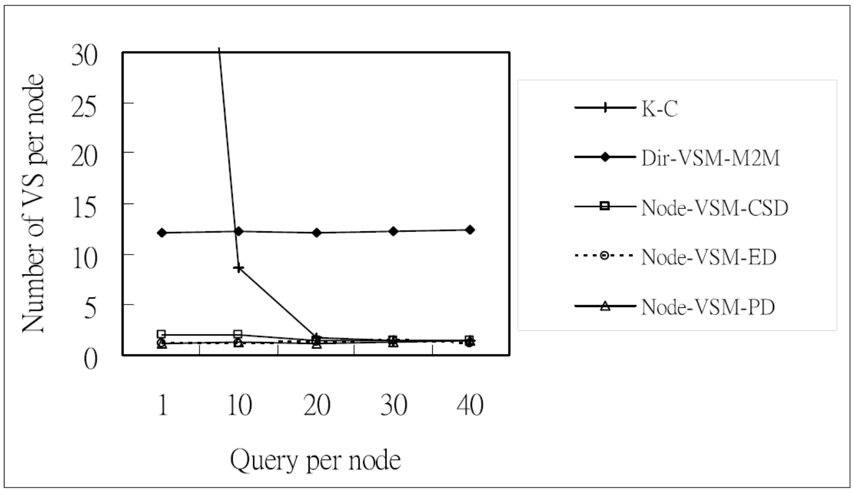

Figure 13. Average number of resulting VS

maintains a relatively stable number of VSs across all query rates, while the average number of VSs of *k-choices* grows to 2 orders of magnitude more than the other algorithms when the query rate is low. The average number of VSs under Dir-VSM-ED, Dir-VSM-CSD and Node-VSM-PD approaches one VS per node on average. Therefore, these two policies effectively transform a VS-based system to have the same level of overhead as non-VS-based systems.

Referring to Figure 14, the average route length of *k-choices* is much higher than those of other algorithms when the query rate is small. The Dir-VSM-ED, Dir-VSM-CSD and Node-VSM-PD have the smallest average route length as a result of the smaller number of VSs in the system. The savings in network bandwidth usage translates into higher lookup success ratios, as will be discussed shortly.

A.4. Success ratio performance of pure discard policies. The query success ratio versus query lookup rate result is plotted in Figure 15. Clearly as query lookup rate increases, the success ratio decreases because of the heavier workload imposed on the system. What is interesting is that having less VSs in the system is good for the performance of the system, as illustrated by the curve for Node-VSM-PD. This is because savings in network bandwidth from reduced stabilization overhead can be used for useful data, resulting in higher query success ratio.

A.5. Resilience to heterogeneous workload conditions. We evaluate the resilience performance of the pure discard VSM policies when the network exhibits heterogeneity in workload, and when the workload may shift dynamically in the network in this section. First, Figure 16 depicts the query success ratio performance of the relevant VSM policies when the workload exhibits degrees of skew in Zipfian distribution, where larger Zipf values denote more skewed workload.

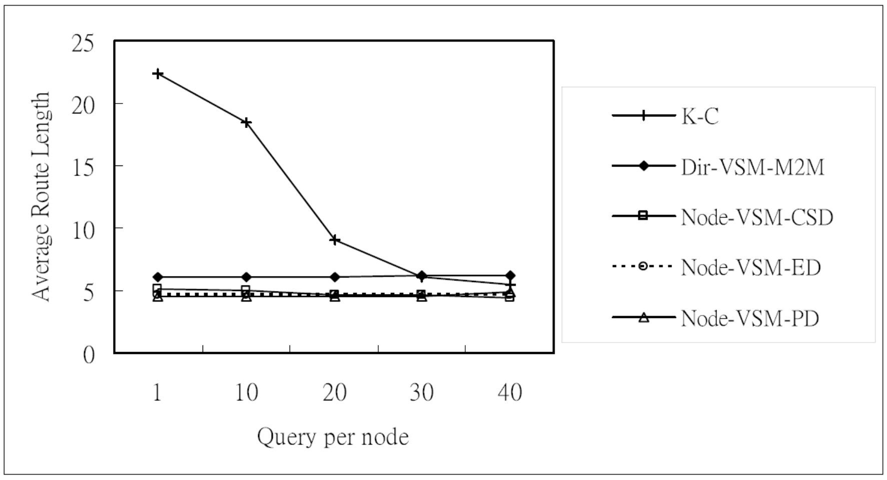

FIGURE 14. Average route length

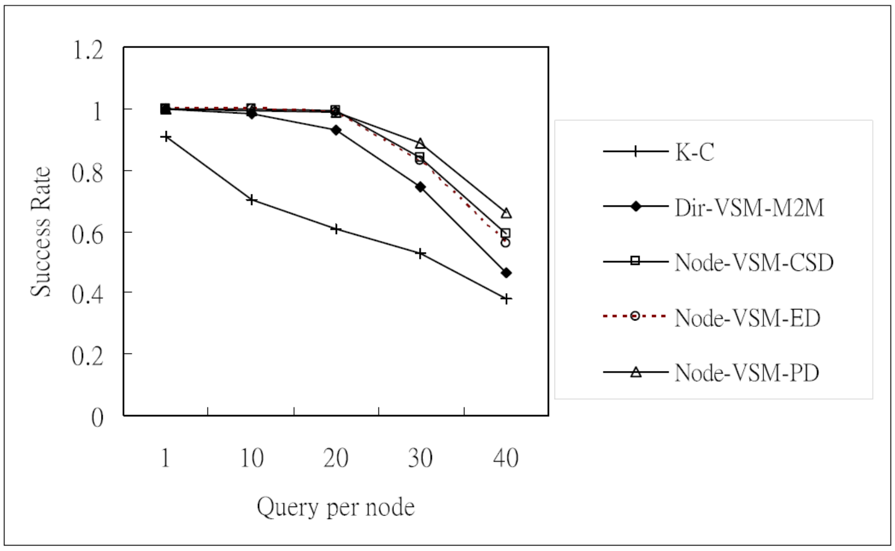

FIGURE 15. Success ratio performance

We first observe that the success ratios decrease as the workload becomes more heterogeneous. This is because more skewed workload causes the query destination to become more concentrated. Therefore, the network bandwidth of the destination node and neighboring nodes becomes more easily exhausted. Secondly, M2M-like systems, including Dir-VSM-M2M, Node-VSM-PD and Node-VSM-ED, achieve higher success ratios than *k-choices* because of the inclusion of the directory nodes, enabling them to find the largest capacity node to sustain the query workload quickly. Finally, Node-VSM-PD achieves the highest success ratio across all workload heterogeneity scenarios, as it retains the advantages of M2M-like systems, while requiring very few VSs.

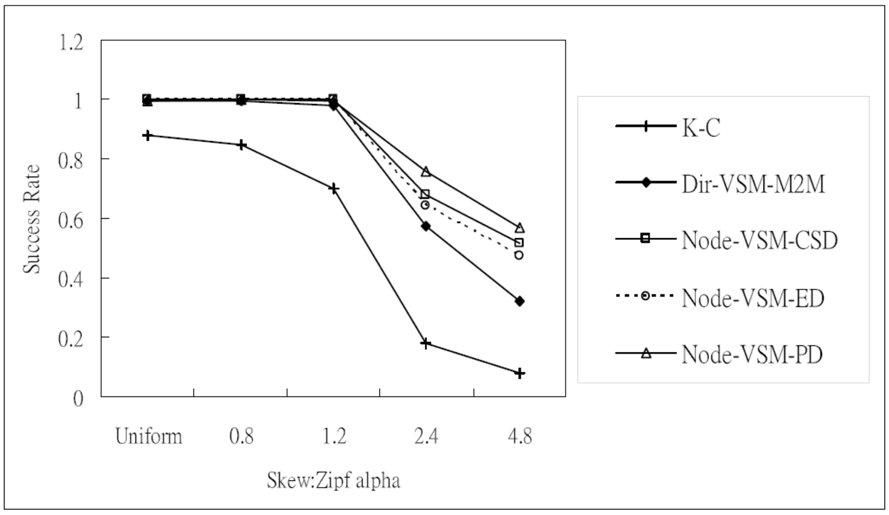

FIGURE 16. Success ratio performance under various degrees of workload heterogeneity

B. Performance of the Adaptive VSM Algorithm. In the previous section, it is see that the pure discard algorithms are not competitive in terms of 99th percentile utilization, due mainly to the minimal number of VSs used in the system. Therefore, when load balance performance is of the highest priority, it is necessary to employ algorithms that can create VSs adaptively according to system utilizations. In this section, we demonstrate that the adaptive VSM algorithm (Node-VSM-Adaptive), can achieve the load balance objective.
B.1. The average number of VSs performance of Dir-VSM-Adaptive. The average number of VSs in the system under Dir-VSM-Adaptive, Node-VSM-PD and Dir-VSM-M2M policies is depicted in Figure 17.

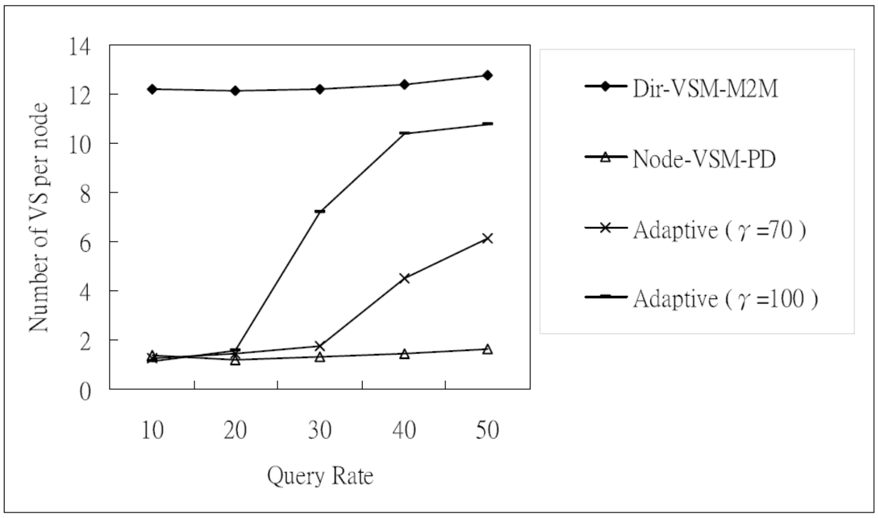

FIGURE 17. Aaverage number of VS vs query rate under Dir-VSM-Adaptive

It can be seen that as query rate increases, the population of VSs also increases under Dir-VSM-Adaptive. The VS creation strategy in Dir-VSM-Adaptive is obviously effective, creating more VSs when needed as system workload grows. In addition, the rate of increase is positively correlated to the value of the γ parameter. Having more VSs is important for load balancing purposes, as more VSs are needed when system utilization is high.
B.2. The load-balance performance of Dir-VSM-Adaptive. Figure 18 depicts the 99th

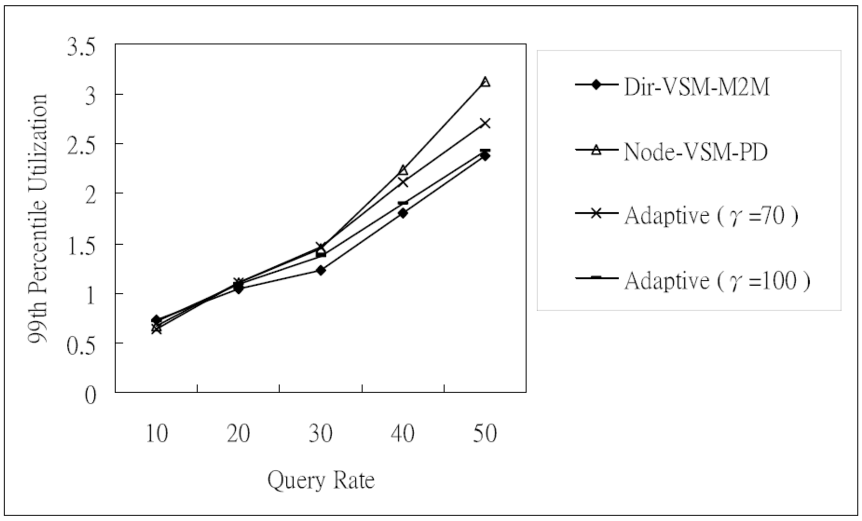

FIGURE 18. 99th percentile utilization vs query rate

percentile utilization results for Dir-VSM-Adaptive, with several parameter settings, and Dir-VSM-M2M. We first observe that, under Dir-VSM-Adaptive, when VSs are adaptively created in more aggressive manner, $\gamma = 70$ or $\gamma = 100$, the 99th percentile utilization improves accordingly across the whole spectrum of query rates, confirming previous finding that more VSs are needed for better load balance. It is interesting to note that Dir-VSM-Adaptive's load balance performance is very close to that of Dir-VSM-M2M, which employs more VSs. It is conceivable that if we had created the VSs in an even more aggressive manner, we would have been able to match the load balance performance of Dir-VSM-M2M. As it is, the performance of Dir-VSM-Adaptive has reached very good load balance performance.

B.3. Query success ratio performance under varying system utilizations. The query success ratio performance results are shown in Figure 19. The results are similar to the pure discard case. Again, as query success is closely related to the number of VSs in the system, it can be seen that systems with fewer VSs outperform those with more VSs. Therefore, Node-VSM-PD performs the best and Dir-VSM-M2M performs the worst. Dir-VSM-Adaptive's performance is between the two extremes, as expected.

C. Impact and Overheads of the Active Stabilization Mechanism. The original Chord stabilization protocol exchanges packets periodically with finger table neighbors to maintain the consistency of finger tables. The stabilization procedure is implemented in the *fix_finger*() and *stabilize*() modules. With the introduction of the RFT mechanism, implemented in $RFT\_manage()$, it is important to understand its benefits and the incurred additional overhead for each node. Note that, since the total number of VSs in the system is reduced in our proposed schemes, the total amount of stabilization overhead should be less than the original M2M's. In this section, we examine the impact of the RFT mechanism on routing state consistency versus query rate, and the incurred additional overheads.

We have also investigated the impact of RFT mechanism in other scenarios, for example, versus node lifetime. The results also confirm the importance of the RFT mechanism. However, those results are not included due to paper length consideration.

C.1. Impact of RFT mechanism on routing state consistency. The impact of the RFT mechanism on routing state consistency is measured in terms of the average percentage of incorrect finger table entries. The results under the relevant VSM algorithms, with and without inclusion of the RFT mechanism versus varying query rate are shown in Figure

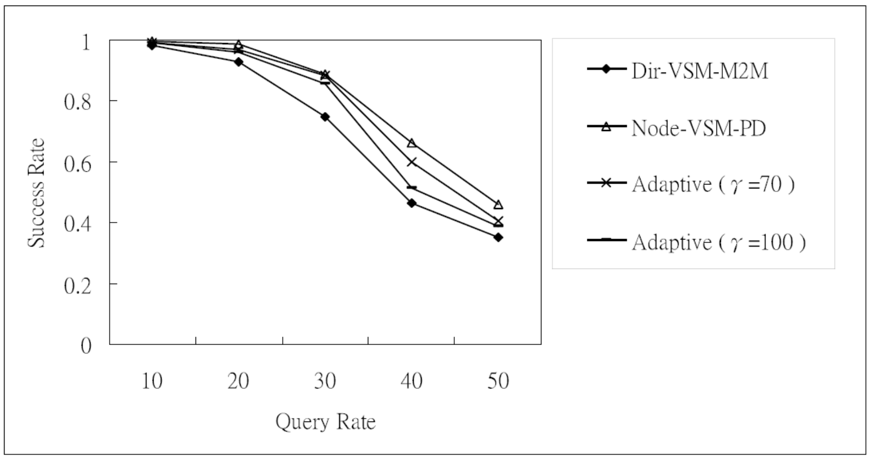

Figure 19. Query success percentage vs query rate

20. The finger table error percentage is obtained by examining the state of the finger tables every simulation second. The average node lifetime is 15 minutes. For Dir-VSM-Adaptive, and when the query rate is 40, the percentage of erroneous finger table entries is less than 8% with the inclusion of the RFT mechanism, and reaches 27% without RFT. The impact on the Node-VSM-PD policy is similar. Therefore, the RFT mechanism is essential to the effectiveness of our proposals, since we create and delete VSs more frequently than the original Chord protocol.

C.2. Stabilization message overhead analysis. The breakdown of the stabilization overhead into *fix_finger*(), *stabilize*() and *RFT_manage*() categories for Node-VSM-PD and Dir-VSM-Adaptive algorithms is shown in Figure 21.

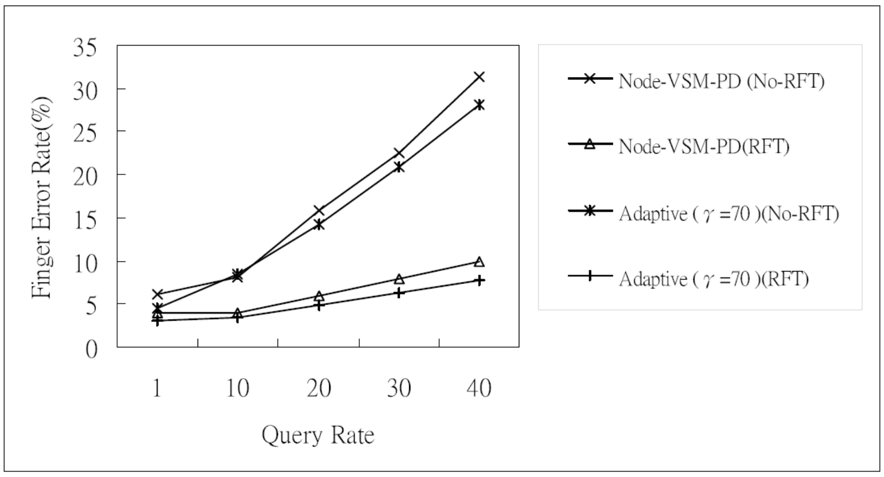

Figure 20. Percentage of incorrect finger table entries with and without RFT

First, note that *RFT_manage*() requires less overhead packets as *fix_finger*() and *stabilize*() when the query rate is low, but requires about the same number of packets as *fix_finger*() and *stabilize*() combined when the query rate is high. Second, when query rate is high, the total stabilization overhead is around only 1% of total packets, which is about an order of magnitude smaller than when query rate is lower. This is because the increase in number of data packets significantly outpaces stabilization packets. Thirdly, the Dir-VSM-Adaptive algorithm requires more overhead packets than Node-VSM-PD when the query rate is high because of larger number of VSs employed. However, the total

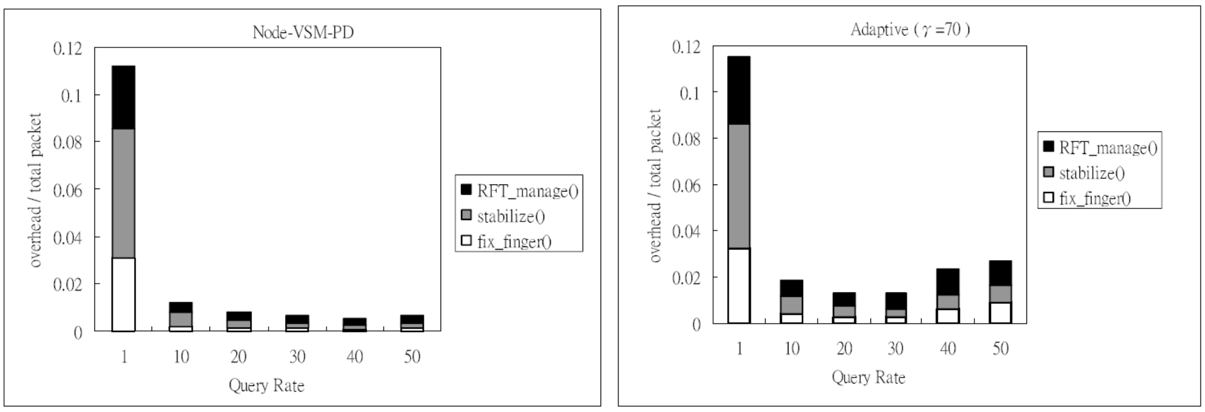

Figure 21. Overhead percentage for Node_VSM_PD, Dir-VSM-Adaptive ($\gamma = 70$) policies

stabilization overhead in both schemes is minimal, while being very effective in maintaining the consistency of the routing state of the system, as demonstrated in previous experiments.

5. **Conclusions and Future Work.**

We have presented a number of virtual server management algorithms whose inclusion into migration-based load balancing schemes makes these schemes truly practical, circumventing the major obstacle faced by these schemes. In particular, we demonstrate the overall effectiveness of the Node-VSM-PD algorithm to achieve good query rate success ratio and acceptable system load balance with minimal overhead, and the power of the Dir-VSM-adaptive algorithm to achieve excellent system workload balance when load balance of utmost importance.

As future work, we intend to investigate the following issues. First, we have assumed that VS migration incurs no costs since the objective for load balance is network bandwidth in this paper. However, the migration cost may be high when the objective for balance is storage objects, as objects may have to moved physically between nodes. This scenario is important in, for example, distributed information retrieval systems, which must store and migrate posting lists [23].

Secondly, with respect to virtual-server management, maintaining a cache of known node capacities by directory nodes and being prudent in VS discard has not proven to be competitive to eager discard policies. This result is expected but the amount of VS reduction is lower than intuitively possible. We plan to investigate this issue further and propose remedies to improve the cache-based schemes as a basis for more prudent approaches for VSM.

Thirdly, the Dir-VSM-adaptive scheme is controlled by a system parameter γ. It provides an indication of how hard the system should try to create VSs in the face of insufficient VSs. However, for a parameter value, the extent to which the workload may be balanced not obvious. It would be nice if users can specify a desired degree of desired load balance, and the system can figure out the proper parameter to use. Methods for automatic determination of the γ parameter setting would be very useful and are a fruitful future research topic.

Finally, we intend to extend our results by including security considerations, based on the works in [4,6,10]. The security issues are not addressed in this paper, but are important when the results in our work are deployed in the real world.

Acknowledgment. This work was supported in part by the National Science Council under contract number NSC-98-2219-E-011-002, NSC-98-2221-E-011-133-MY3, NSC-98-2923-E-011-004-MY3, NSC-99-2916-I-011-002-A1 and NSC 100-2221-E-011-123.

REFERENCES

[1] P. B. Godfrey and I. Stoica, Heterogeneity and load balance in distributed hash tables, *Proc. of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies*, vol.1, pp.596-606, 2005.

[2] P. B. Goedfrey, S. Shenker and I. Stoica, Minimizing churn in distributed systems, *Proc. of ACM SIGCOMM*, Pisa, Italy, 2006.

[3] D. Boukhelef and H. Kitagawa, Dynamic load balancing in RCAN content addressable network, *Proc. of the 3rd International Conference on Ubiquitous Information Management and Communication*, 2009.

[4] C.-C. Chang, J.-S. Lee and J.-Y. Kuo, Time-bound based authentication scheme for multi-server architecture, *International Journal of Innovative Computing, Information and Control*, vol.4, no.11, pp.2987-2996, 2008.

[5] C. Chen and K.-C. Tsai, The server reassignment problem for load balancing in structured P2P systems, *IEEE Transactions on Parallel and Distributed Systems*, vol.12, no.2, pp.234-246, 2008.

[6] C.-H. Chen, A practical voting scheme using one server with a secure coprocessor, *ICIC Express Letters*, vol.3, no.3(B), pp.585-592, 2009.

[7] F. Dabek, F. Kaashoek, D. Karger, R. Morris and I. Stoica, Wide-area cooperative storage with CFS, *Proc. of ACM SOSP*, Banff, Canada, 2001.

[8] H.-C. Hsiao, H. Liao, S.-T. Chen and K.-C. Huang, Load balance with imperfect information in structured peer-to-peer systems, *IEEE Transactions on Parallel and Distributed Systems*, vol.22, no.4, pp.634-649, 2010.

[9] J. Jiang, F. Tang, F. Pan and W. Wang, Using bidirectional links to improve peer-to-peer lookup performance, *Journal of Zhejiang University Science A*, vol.7, no.6, pp.945-951, 2006.

[10] J.-S. Lee, Y.-F. Chang and C.-C. Chang, A novel authentication protocol for multi-server architecture without smart cards, *International Journal of Innovative Computing, Information and Control*, vol.4, no.6, pp.1357-1364, 2008.

[11] D. 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 wide web, *ACM STOC*, pp.654-663, 1997.

[12] J. Ledlie and M. Seltzer, Distributed, secure load balancing with skew, heterogeneity, and churn, *Proc. of IEEE INFOCOM*, pp.1419-1430, 2005.

[13] J. Li, J. Stribling, R. Morris, M. F. Kaashoek and T. M. Gil, A performance vs. cost framework for evaluating DHT design tradeoffs under churn, *Proc. of IEEE INFOCOM*, 2005.

[14] S. Rhea, D. Geels, T. Roscoe and J. Kubiatowicz, Handling churn in a DHT, *Proc. of the USENIX Annual Technical Conference*, 2004.

[15] A. Rao, K. Lakshminarayanan, S. Surana, R. Karp and I. Stoica, Load balancing in structured P2P systems, *IPTPS*, Berkeley, CA, USA, 2003.

[16] G. Skobeltsyn, T. Luu, I. P. Zarko, M. Rajman and K. Aberer, Web text retrieval with a P2P query-driven index, *Proc. of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Amsterdam, The Netherlands, 2007.

[17] E. Sit, A. Haeberlen, F. Dabek, B.-G. Chun, H. Weatherspoon, R. Morris, M. F. Kaashoek and J. Kubiatowicz, Proactive replication for data durability, *Proc. of the 5th International Workshop on Peer-to-Peer Systems*, 2006.

[18] I. Stoica, R. Morris, D. Liben-Nowell, D. R. Karger, M. F. Kaashoek, F. Dabek and H. Balakrishnan, Chord: A scalable peer-to-peer lookup protocol for internet applications, *IEEE/ACM Transactions on Networking*, vol.11, no.1, pp.17-32, 2003.

[19] S. Surana, B. Godfrey, K. Lakshminarayanan, R. Karp and I. Stoica, Load balancing in dynamic structured P2P systems, *Performance Evaluation*, vol.63, no.3, pp.217-240, 2006.

[20] K. Tati and G. M. Voelker, On object maintenance in peer-to-peer systems, *Proc. of the 5th International Workshop on Peer-to-Peer Systems*, 2006.

[21] X. Wang and D. Loguinov, Load-balancing performance of consistent hashing: Asymptotic analysis of random node join, *IEEE/ACM Transactions on Networking*, vol.15, no.5, 2007.

[22] D. Wu, Y. Tian and K.-W. Ng, On the effectiveness of migration-based load balancing strategies in DHT systems, *Proc. of the 15th International Conference on Computer Communications and Networks*, Arlington, VA, pp.405-410, 2006.

[23] Y. Yang, R. Dunlap, M. Rexroad and B. F. Cooper, Performance of full text search in structured and unstructured peer-to-peer systems, *Proc. of the 25th IEEE International Conference on Computer Communications*, Barcelona, Spain, pp.1-12, 2006.

[24] A. Law and W. D. Kelton, *Simulation Modeling and Analysis*, 3rd Edition, McGraw-Hill, 1999.

[25] *P2psim: A Simulator for Peer-to-Peer Protocols*, http://pdos.csail.mit.edu/p2psim/.

[26] S.-J. Horng, M.-Y. Su, Y.-H. Chen, T.-W. Kao, R.-J. Chen, J.-L. Lai and C. D. Perkasa, A novel intrusion detection system based on hierarchical clustering and support vector machines, *Expert Systems with Applications*, vol.38, no.1, pp.306-313, 2011.

[27] S.-J. Horng, P. Fan, Y.-P. Chou, Y.-C. Chang and Y. Pan, A feasible intrusion detector for recognizing IIS attacks based on neural networks, *Computers and Security*, vol.27, no.3-4, pp.84-100, 2008.

[28] S.-J. Horng, P. Fan, M.-Y. Su, Y.-H. Chen, C.-L. Lee and S.-W. Lan, Anomaly detection for web server based on smooth support vector machine, *Computer Systems Science and Engineering*, vol.23, no.3, pp.209-218, 2008.

[29] Y.-P. Chou, S.-J. Horng, H.-Y. Gu, C.-L. Lee, Y.-H. Chen and Y. Pan, Detecting pop-up advertisement browser windows using support vector machines, *Journal of the Chinese Institute of Engineers*, vol.31, no.7, pp.1189-1198, 2008.

[30] C. Chen, C.-L. Tsai and S.-J. Horng, Exploiting attribute popularity distribution skew to enhance the performance of peer to peer publish/subscribe systems, *International Journal of Innovative Computing, Information and Control*, vol.7, no.7(A), pp.4047-4066, 2011.