RESEARCH ARTICLE OPEN ACCESS

Preserving Interflow Packet Order in Multipath Switching Systems Using Flow-Based Slicing For Congestion Avoidence

A. Venkata Pradeep*, M. Kishore Kumar**
*(M.Tech Student, Department of Computer Science, Vivekananda Institute of Technology & Science, Housing Board Colony, By-Pass Road, Karimnagar , AP, India)
** (Associate Professor, Department of Computer Science & Engineering, Vivekananda Institute of Technology & Science, Housing Board Colony, By-Pass Road, Karimnagar , AP, India)

ABSTRACT
Congestion avoidance is an important traffic engineering task, which can be performed by making use of load-balancing technique when a link is over loaded or failure of link occur in Multi-Path Switching Systems . Multipath Switching Systems are capable of transferring high rate data in networks. As preserving the inter flow packet orders while avoiding congestion is a major issue in consideration and previous packet based solutions require packet reordering which cause delay penalties. In this paper we use a flow-based slicing scheme which slits each flow in to slices at every inter- flow interval larger than the slicing threshold and balances the load to a finer granularity. We depict that flow-based slicing achieves inter flow packet ordering with little network cost while minimizing the packet out of order probability to negligible level comparatively less than 10^{-6}.
Keywords - Congestion Avoidence, Flow-Based Slicing(FBS), Interflow Packet Ordering, Multipath Switching Systems(MPS),

I. INTRODUCTION

Congestion avoidance is an important traffic engineering task. Our specific aim is to minimize the maximum load in Multipath Switching System which play vital role in fabricating the state-of- art high performance core routers while preserving the interflow packet orders, that is the packets in the same flow should depart as that of their arrival orders at multipath switching systems along with uniform load sharing to avoid congestion and with low complexity.

Load balancing is defined as the allocation of the work of a single application to processors at runtime so that the execution time of the application is minimized. The optimal load balancing policy is developed and extended to develop a distributed load-balancing policy that can dynamically reallocate incoming external loads at each node.

The two proposed approaches in considerably improving load balancing and extending network lifetime

- Load balancing in two-node distributed system.
- Load balancing using Regeneration Theory.

In packet-based solutions, the traffic is dispatched packet by packet to optimally to avoid congestion problem. However, in this packets in the same flow may be forwarded in the separate paths and experience various delays, thus violating the intra-flow packet ordering requirement. Although timestamp or sequence based re-sequencers can be added to restore packet orders, they are often shown to be costly and not scalable. By timestamp based re-sequencer, each packet is slowed down statically (or adaptively) by the system delay upper bound, which will impose a huge delay penalty. On the other hand, the sequence based re-sequencer will need to maintain at least N re-sequencers at each output, leading to $O(N2)$ complexity. To avoid the packet out-of-order, another choice is to use flow-based traffic-balancing algorithms.

Here it dispatches packets in the same flow to a fixed switching path by hashing its 5-tuple to path ID.

However, hashing solution will lead to severe load-imbalance. In this paper, we present a new scheme, namely *Flow-Based Slicing* that perfectly achieves the three objectives defined above. Here the intra-flow packet intervals are often; say in 40-50 percentages, larger than the delay upper bound at MPS which is calculated statistically. As such, if we cut off each flow at every packet interval larger than a flow-cut threshold equaling to this bound and balance the load on the generated flow-based slices, the three objectives are met triply:

1. The traffic-balancing uniformity of is only moderately degraded from the optimal traffic-balancing.
2. The intra-flow packet order is kept intact as their arrivals. Exceptions only happen in a negligible level (10^{-6}).
3. The flow-based slices table size to implement FBS only requires 1.8MB under 40Gbps line rate, which can be placed on- chip to provide an ultra-fast access speed.

II. LITERATURE SURVEY

The most important step in software development process is Literature survey. Determining the time factor, economy and company strength is necessary prior developing the tool. Once these things are satisfied, then next steps is to determine which operating system and language can be used for developing the tool. Once the programmers start building the tool the programmers need lot of external support. This support can be obtained from senior programmers, from book or from websites. Before building the system the above consideration are taken into account for developing the proposed system.

Flow-Based Slicing

A flow-based slicing is nothing but sequence of packets in a flow, where every intra flow interval between two consecutive packets is greater than or equal to a slicing threshold Φ. Flow-based slices can be seen as mini flows created by cutting off every intra flow interval larger than Φ. Flow-based slicing scheme can be described as follows. The flow-based slicing scheme is applied to all the nodes other than the source node. When the source node receives the data it first identifies all the possible paths to reach the destination. Out of all the possible paths identified it only identifies such path which does not have shared link. Depending on the number of paths without shared link the data at the source will be divided and sent across those selected paths. Once each node receives the data it slices the data. The data sliced depends upon the bandwidth of the node and the slicing threshold value. At each node, the sliced data is kept within itself and the remaining data is sent to its neighbor. The same process is repeated until the data is reached to the destination. At the destination only some part of the data is reached. The remaining data which is present at each node as a result of slicing will reach the destination through buffering concept. The advantages of flow slice are:

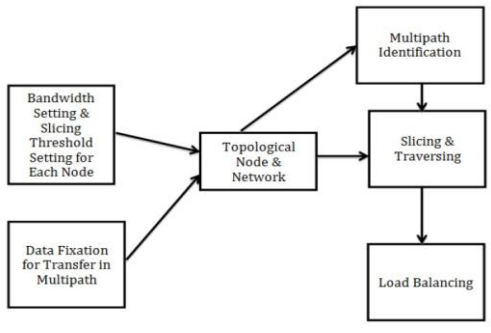

Fig 2.1 General Design of Flow-Based Slice Concept

1. It is immune to packet loss, while other solutions like the VIQ re sequencer require additional loss detection mechanisms.
2. It maintains a hash table to record active flow-slice context, a redirection mechanism can be added to provide robustness to system failure. When some switching path stops working, the load balancer can simply redirect all the active flow slices going to this path at their next packet arrivals, still by FS.
3. It natively supports multicast. Multicast flows are treated in the same manner as uni-cast flows and still preserve packet orders.
4. N-FS supports load balancing across uneven switching paths by applying weighted round robin.

Features of Flow-Based Slicing

- **Small average packet size.**

Both the average packet count (FC) and the average size (FS) of FBS are much smaller than those of the original flow.

- **Light-tailed size distribution.**

FBS packet count/size distributions are light tailed while it is well-known that original flow-size distribution is heavy tailed.

- **Fewer active flow-slices**

The active FBS number is 1-2 magnitudes smaller than that of active flow.

Objectives of Multipath Switching System

One among the major open issues in MPS is preserving the interflow packet order as how to distribute incoming traffic A(t) across its k internal switching paths $\{T_l\}(l \in [1,k])$ to meet at least three objectives simultaneously:

- Uniform load sharing.
- Intraflow packet ordering.
- Low timing and hardware complexity.

1. Uniform load sharing

Traffic dispatched to each path should be uniform. Specifically in MPS, traffic destined for each output should be spread evenly to avoid output contention, minimize average packet delay, and maximize throughput. This requirement is formalized as

$$\textbf{Equalize } \{A^l_j(t)\} \ (l \in [1,k]) \textbf{ for any } j$$

where $A^l_j(t)$ denotes the traffic rate destined for output port j through switching path l in MPS.

2. Intraflow packet ordering

Packets in the same flow should depart MPS as their arrival orders. This ordering is essential since out-of- order packets will degrade the performance of higher level protocols. For any two packets P_1 and P_2 in the same flow with arrival time $T(P_1)$, $T(P_2)$, and departure time $D(P_1)$, $D(P_2)$, the formula below should be guaranteed:

$$\textbf{If } T(P_1) < T(P_2) \textbf{ then } D(P_1) < D(P_2)$$

3. Low timing and hardware complexity

The load-balancing and additional resequencing mechanisms at MPS should work fast enough to match the line rate, and should introduce limited hardware complexity. MPS is most likely to hold hundreds of external ports operating at ultrahigh

speed. To provide such scalability, the timing/hardware complexity of O(1) is necessary.

As a rule of thumb, packet-based solutions are advocated where traffic is dispatched packet by packet to optimally balance the load. However, packets in the same flow may be forwarded in separate paths and experience different delays, thus violating the interflow packet ordering requirement. A straightforward solution is to use an explicit resequencer at each output to restore packet orders.

Each packet is time shifted by the same offset before departing, thus preserving the arrival order. Nonetheless, the delay equalization method suffers from a huge penalty in magnifying the average delay. It is shown in our prototype simulations that even the latest adaptive resequencer increases the average delay nearly 10 times to about 10 ms. A route passing through five such routers will lead to at least 50 ms average delay, almost violating the QoS requirement for delay-sensitive applications. Another kind of resequencers records each packet's sequence number in the flow (defined by input, output port, and priority class), instead of absolute timestamp. By allowing only in-order packets with expected sequence number to depart, they preserve packet orders without penalizing packet delays.

III. SYSTEM ARCHITECTURE

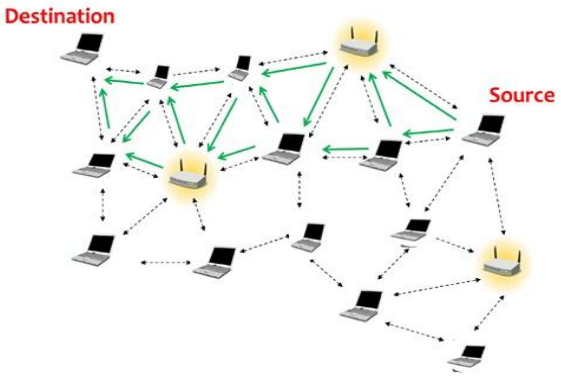

Fig 3.1 Architecture of Multipath Switching System

Existing System

Our major improvement over the existing works is to tailor the approach in the scenario by introducing the offline delay bound calculation, while the previous solutions either use an empirical slicing threshold or maintain flow context to facilitate the slicing. The traces here are collected at backbone links of one of the largest commercial backbones worldwide.

Disadvantage

Our major improvement over the existing works is to tailor the FS approach in the MPS scenario by introducing the offline delay bound.

Problem Definition

Packets in the same flow are order by using flow-based load-balancing algorithms. MPS is most likely to hold hundreds of external ports operating at ultrahigh speed. To provide such scalability, the timing/hardware complexity of O(1) is necessary. packets in the same flow may be forwarded in separate paths and experience different delays, thus violating the intraflow packet ordering requirement .

Disadvantages

- Low timing and hardware complexity.
- Intraflow packet ordering
- packet-based solutions either suffer from delay penalties.

Proposed System:

Flow-Based Slicing is based on the fact that the interflow packet interval is often, larger than the slicing threshold. Due to three positive properties of flow-based slicing, our scheme achieves good load-balancing uniformity with little hardware overhead and timing complexity. By calculating delay bounds at three popular MPS, we show that when the slicing threshold is set to the smallest admissible value at, the FBS scheme can achieve optimal performance while keeping the interflow packet out-of-order probability negligible given an internal speedup up to two. Our results are also validated through trace-driven prototype simulations under traffic patterns.

Advantage

It is immune to packet loss, while other solutions like the resequencer require additional loss detection Mechanisms.

IV. MODULES

1) Load-Balancing Scheme

Interflow packet order is natively preserved besetting slicing threshold to the delay upper bound at MPs .Any two packets in the same flow-based slice cannot be disordered as they are dispatched to the same switching path where processing is guaranteed; and two packets in the same flow but different flow - based slices will be in order at departure, as the earlier packet will have depart from before the latter packet arrives. Due to the fewer number of active flow slices, the only additional overhead in, the hash table, can be kept rather small, and placed on-chip to provide ultrafast access speed. This table size depends only on system line rate and will stay unchanged even if scales to more than thousand external ports, thus guarantees system scalability.

2) Multipath Switching System

Through lay-aside Buffer Management module, all packets are virtually queued at the output according to the flow group and the priority class in a hierarchical manner. The output scheduler fetches packets to the output line using information provided . Packets in the same flow will be virtually buffered in the same queue and scheduled in discipline. Hence, intraflow packet departure orders hold as their arriving

orders at the multiplexer. Central-stage parallel switches adopt an output-queued model. By Theorem, we derive packet delay bound at first stage. We then study delay at second-stage switches. Define native packet delay at stage m of an be delay experienced at stage m on the condition that all the preceding stages immediately send all arrival packets out without delay.

3) Multistage Multi plane Clos Switches

We consider the Multistage Multi plane Clos network based switch by Chao et a . It is constructed of five stages of switch modules with top-level architecture similar to a external input/output ports.

The first and last stages clos are composed of input de multiplexers and output multiplexers, respectively, having similar internal structures as those in PPS. Stages 2-4 of M^2Clos are constructed by parallel switching planes; however, each plane is no longer formed by a basic switch, but by a three-stage Clos Network to support large port count. Inside each Clos Network, the first stage is composed by k identical Input Modules.

Each IM is a packet switch, with each output link connected to a Central Module. Thus, there is a total of m identical in second stage of the Clos networks.

V. SOFTWARE ENVIRONMENT

Features OF. Net

Microsoft .NET is a set of Microsoft software technologies for rapidly building and integrating XML Web services, Microsoft Windows-based applications, and Web solutions. The .NET Framework is a language-neutral platform for writing programs that can easily and securely interoperate.

There's no language barrier with .NET: there are numerous languages available to the developer including Managed C++, C#, Visual Basic and Java Script. The .NET framework provides the foundation for components to interact seamlessly, whether locally or remotely on different platforms. It standardizes common data types and communications protocols so that components created in different languages can easily interoperate.

".NET" is also the collective name given to various software components built upon the .NET platform. These will be both products (Visual Studio.NET and Windows.NET Server, for instance) and services (like Passport, .NET My Services, and so on).

The .Net Framework

The .NET Framework has two main parts:

1. The Common Language Runtime (CLR).
2. A hierarchical set of class libraries.

The CLR is described as the "execution engine" of .NET. It provides the environment within which programs run. The most important features are

- Conversion from a low-level assembler-style language, called Intermediate Language (IL), into code native to the platform being executed on.
- Memory management, notably including garbage collection.
- Checking and enforcing security restrictions on the running code.
- Loading and executing programs, with version control and other such features.
- The following features of the .NET framework are also worth description:

Managed Code

The code that targets .NET, and which contains certain extra Information - "metadata" - to describe itself. Whilst both managed and unmanaged code can run in the runtime, only managed code contains the information that allows the CLR to guarantee, for instance, safe execution and interoperability.

Managed Data

With Managed Code comes Managed Data. CLR provides memory allocation and Deal location facilities, and garbage collection. Some .NET languages use Managed Data by default, such as C#, Visual Basic.NET and JScript.NET, whereas others, namely C++, do not. Targeting CLR can, depending on the language you're using, impose certain constraints on the features available.

As with managed and unmanaged code, one can have both managed and unmanaged data in .NET applications - data that doesn't get garbage collected but instead is looked after by unmanaged code.

Common Type System

The CLR uses something called the Common Type System (CTS) to strictly enforce type-safety. This ensures that all classes are compatible with each other, by describing types in a common way.

CTS define how types work within the runtime, which enables types in one language to interoperate with types in another language, including cross-language exception handling. As well as ensuring that types are only used in appropriate ways, the runtime also ensures that code doesn't attempt to access memory that hasn't been allocated to it.

Common Language Specification

The CLR provides built-in support for language interoperability. To ensure that you can develop managed code that can be fully used by developers using any programming language, a set of language features and rules for using them called the Common Language Specification (CLS) has been defined. Components that follow these rules and expose only CLS features are considered CLS-compliant.

The Class Library

.NET provides a single-rooted hierarchy of classes, containing over 7000 types. The root of the namespace is called System; this contains basic types like Byte, Double, Boolean, and String, as well as Object. All objects derive from System. Object. As well as objects, there are value types. Value types can be allocated on the stack, which can provide
useful flexibility. There are also efficient means of converting value types to object types if and when necessary. The set of classes is pretty comprehensive, providing collections, file, screen, and network I/O, threading, and so on, as well as XML and database connectivity.

The class library is subdivided into a number of sets (or namespaces), each providing distinct areas of functionality, with dependencies between the namespaces kept to a minimum.

VI. EXPERIMENTAL RESULTS

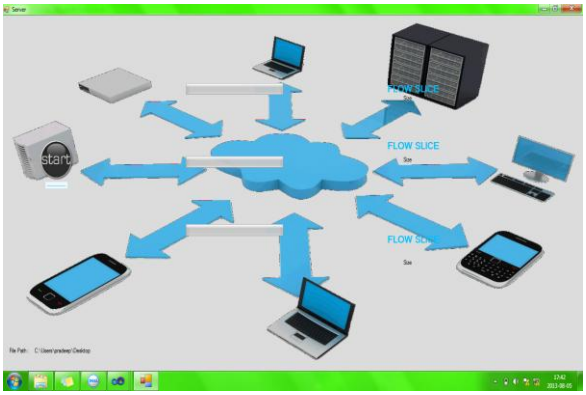

Fig 6.1 Sever Initiation

Here three clients along with their file size are displayed and respective progress bar shows the file transmission status.

Whenever Server window is loaded updates the file receiving path at right side corner of window. Click on the picture box 'Start' label below it displays 'Server Running' i.e., Server is ready to do load balancing by accepting inputs and is in running state.

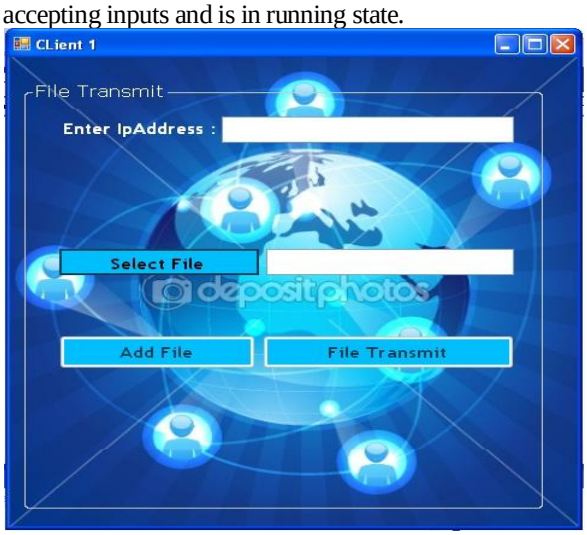

Fig 6.2 Client1 Initiation

It consists of a text box labled 'Enter IpAddress', by selecting a file from the bowser widow which appear by clicking 'Select File' button, client1 will add file to transmit it to destination.

We will select a file from browser window and add file by clicking 'Add File' button. Click on button 'File Transmit' for transmission of file. Similarly Client2, Client 3 are initiated.

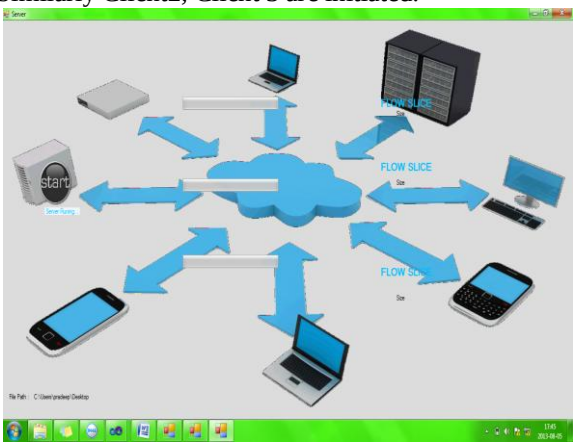

Fig 6.3 Server in Running State

Click on the picture box 'Start' label below it displays 'Server Running' i.e., Server is ready to do load balancing by accepting inputs and is in running state.

Whenever we obtain 'Server Running..' at label below the 'Start' button it means that a Back Ground Worker is running behind which sets the timer control ON and the timer control interval is given as1500ms for the progress bar.

Fig 6.4 File uploading at Client1for File Transmitt

In general, we enter the default Ip Address 127.0.0.1 otherwise we can also give the Ip Address of the system in which the code is executing. Here Ip Address is nothing but Destination Address i.e., Server Ip Address. We will select a file from browser window and add file by clicking 'Add File' button. Click on button 'File Transmit' for transmission of file. Similarly we will upload and transmit file at Client2, Client3 respectively.

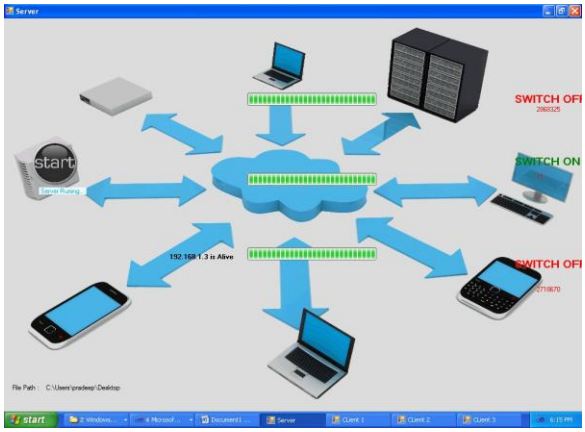

Fig .6.7 Completion of file transmission using Flow-Based Slicing while preserving the inter flow packet order and avoiding congestion.

When Server is in running state and when file is uploaded successfully size of the file is read first and a label below Flow Slice shows the size of the file, then progress bar shows the flow-based slicing process status for respective nodes which is active and the respective input multiplexer and output demultiplexers of M^2Clos switch are set to condition 'SWITCH ON'. In timer control interval for the progress bar is given as 1500ms. After Completion of File Transmit we can see the complete file at destination mean while we can observe that Slices are obtained at respective destination folders.
When the progress bar is complete file is transmitted to destination path which is shown at right corner of the server window. Here Client 1,2,3 files are received at destination folders MS Client 1, MS Client 2, MS Client 3 on Desktop.

VII. CONCLUSION

We proposed a flow-based slicing scheme for preserving interflow packet order by setting slicing threshold to the delay upper bound at MPS .Any two packets in the same flow-based slice cannot be disordered as they are dispatched to the same switching path where processing is guaranteed; and two packets in the same flow but different flow -based slices will be in order at departure, as the earlier packet will have depart from before the latter packet arrives. It is shown that flow-based slicing achieves inter flow packet ordering with little network cost while minimizing the packet out of order probability to negligible level comparatively less than 10-6.

REFERENCES

[1] GAGANA. K AND ASHA "TOPOLOGY CONTROL ACROSS MPS USING FLOW SLICE IN WIRELESS NETWORKS", *INTERNATIONAL JOURNAL OF ENGINEERING AND SCIENCE VOL.3, ISSUE 3 (JUNE 2013), PP 37-41 ISSN(E): 2278-4721, ISSN (P):2319-6483.*

[2] Srikanth Kandula, Dina Katabi, Shantanu Sinha and Arthur Berger "Dynamic Load Balancing Without Packet Reordering", *ACM SIGCOMM Computer Communication Review, Volume 37, Number 2, April 2007*

[3] Lei Shi, Bin Liu, Changhua Sun, Zhengyu Yin, Laxmi N. Bhuyan, H. Jonathan Chao, "Load-Balancing Multipath Switching System with Flow Slice", *IEEE Transactions on Computers, Vol. 61, No. 3, March 2012.*

[4] Nikolaos, I. Chrysos "Congestion Management for Non-Blocking Clos Networks" *Institute of Computer Science, ICS, Forth Hellas http://archvlsi.ics.forth.gr.*

[5] Weiguang Shi, M. H. MacGregor, "Load Balancing for Parallel Forwarding", *IEE/ACM Transactions On Networking, Vol. 13, No. 4, August 2005.*

[6] J. S. Turner, "Resequencing Cells in an ATM Switch", *Tech. Rep., WUCS-91-21, Feb. 1991.*

[7] D. A. Khotimsky and S. Krishnan, "Evaluation of Open-loop Sequence Control Schemes for Multi-path Switches," *in Proc. IEEE ICC, pp. 2116-2120, 2002.*

[8] L. Shi, W. Li, B. Liu, and X. Wang, "Flow Mapping in the Load Balancing Parallel Packet Switches", *in Proc. IEEE HPSR, pp. 254-258, 2005.*

[9] Cisco CRS-1, *http://www.cisco.com/go/crs/, 2011.*

A. Venkata Pradeep, received Bachelor of Technology degree in Computer Science and Engineering from Vivekananda Institute of Technology & Science (n9), Jawaharlal Technological University, Hyderabad, India in 2010, Currently pursuing Master of technology degree in Computer Science from Vivekananda Institute of Technology & Science(n9), Jawaharlal Technological University, Karimnagar, AP, India. His intense zeal in networks included the research interests in Networking, Ant Net and Traffic Engineering Tasks.

M.Kishore Kumar received his B.Tech degree from Jawaharlal Nehru Technology University, Hyderabad, in 2006 and his Master degree in Software Engineering from Jawaharlal Nehru Technology University, Hyderabad, in 2010. He was an assistant professor in the Department of Computer Science and Engineering at Dr.V.R.K college of Engineering from July 2006 to May 2008. He then joined the Department of Computer Science and Engineering at Vivekananda Institute of Technology and Science in May 2008 as an assistant professor. His research interests include spatial data mining, mobile computing, and network security.