ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

Consignment Stability Multi-Channel Trade System with Dribble Portion

Nandam Pavan Kumar
M.Tech Student,
Nimra College of Engineering and Technology.

Khamar zahan
Guide,
Nimra College of Engineering and Technology.

ABSTRACT:

Multipath Switching systems are intensely used in state-of-the-art core routers to provide terabit or even petabit switching capacity. One of the most intractable issues in designing MPS is ow to load balance traffic across its multiple paths while not disturbing the inter-flow packet orders. Previous packet-based solutions either suffer from delay penalties or lead to hardware complexity, hence do not scale. Flow-based hashing algorithms also perform badly due to the heavy-tailed flow-size distribution. In this paper, we develop a novel scheme, namely, Flow Slice that cuts off each flow into flow slices at every interflow interval larger than a slicing threshold and balances the load on a finer granularity.

Based on the studies of tens of real Internet traces, we show that setting a slicing threshold of 1 _4 ms, the FS scheme achieves comparative load-balancing performance to the optimal one. It also limits the probability of out-of-order packets to a negligible level on three popular at the cost of little hardware complexity and an internal speedup up to two. These results are proven by theoretical analyses and also validated through trace-driven prototype simulations. In this project a novel load-balancing scheme, namely, Flow Slice, based on the fact that the intraflow packet interval is often, larger than the. Due to three positive properties of flow slice, our scheme achieves good load-balancing uniformity with little hardware overhead and timing complexity. By calculating delay bounds at three popular, we show that when the slicing threshold is set to the smallest admissible value at, the FS scheme can achieve optimal performance while keeping the intra-flow 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.

INTRODUCTION:

MULTIPATH Switching systems (MPS) play a pivotal role in fabricating state-of-the-art high performance core routers. A well-known paradigm is the deployment of Benes multistage switches in Cisco CRS-1. Other examples include the Vitesse switch chip family implementing the Parallel Packet Switch (PPS), and the Load-balanced Birkh- off-von Neumann (LBvN) switches. In general, MPS is built by aggregating several lower speed switches and, therefore, exhibits multiple internal data paths.One major open issue in MPS is the load-balancing problem defined as how to distribute incoming traffic A(t) across its k internal switching paths{Tl} (l€[1,k]) to meet at least three objectives are1. 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 formalizedas.

$$\text{Equalize } \{A_j^l(t)\}(l \in [1,k]) \text{ for any } j,$$

Where Alj (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. (Unless otherwise stated, flow in this paper is defined by TCP/IP 5-tuple.) This ordering is essential since out-of-order packets will degrade the performance of higher level protocols. For any two packets P1 and P2 in the same flow with arrival time T(P1), T(P2), and departure time D(P1), D(P2), the formula below should be guaranteed:

$$D(P_1) < D(P_2) \text{ if } T(P_1) < T(P_2).$$

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

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 operating at ultrahigh speed. To provide such scalability, the timing/hardware complexity of O(1) is necessary. 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 intraflow packet ordering requirement. A straight forward solution is to use an explicit resequencer at each output to restore packet orders.

Previous systems cannot gracefully deal with the load-balancing problem in MPS to meet the three objectives outlined above. Here develop a new scheme called Flow Slice (FS) that achieves our load- balancing goals perfectly. Based on the observations on tens of broadly located Internet traces, that the intraflow packet intervals are often, say in 40-50 percent, larger than the delay upper bound at MPS which can be calculated statistically. Flow slice cut off each flow at every interval larger than a slicing threshold set to this bound and balance the load on the generated flow slices.

The major improvement over the existing works is to tailor the FS approach in the MPS scenario by introducing the offline delay bound calculation, while the previous solutions either use an empirical slicing threshold, e.g., 60 ms in, or maintain flow context to facilitate the slicing The empirical slicing threshold will lead to poor load-balancing performance for MPS. Maintaining flow context is impractical in the MPS scenario as it requires a flow table for each input updated by all the outputs in real time.

2. FEASIBILITY STUDY:

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company.

For feasibility analysis, some understanding of the major requirements for the system is essential.

a) Economical feasibility:

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

b) Technical feasibility:

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

c) Social feasibility:

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

3. SOFTWARE DEVELOPMENT LIFE CYCLE:

The software development life cycle (SDLC),also referred to as the application development life-cycle, is used in systems engineering, information systems and software engineering to describe a process for planning, creating, testing, and deploying an information system.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

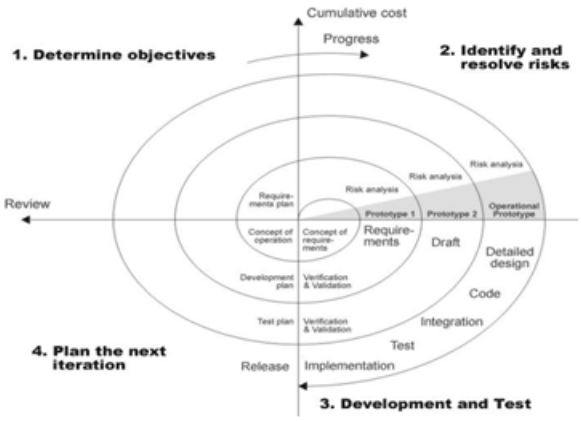
The systems development life-cycle concept applies to a range of hardware and software configurations, as a system can be composed of hardware only, software only, or a combination of both. SPIRAL MODEL was defined by Barry Boehm in his 1988 article, "A spiral Model of Software Development and Enhancement. This model was not the first model to discuss iterative development, but it was the first model to explain why the iteration models.

As originally envisioned, the iterations were typically 6 months to 2 years long. Each phase starts with a design goal and ends with a client reviewing the progress thus far. Analysis and engineering efforts are applied at each phase of the project, with an eye toward the end goal of the project.

The steps for Spiral Model can be generalized as follows:

a) The new system requirements are defined in as much details as possible. This usually involves interviewing a number of users representing all the external or internal users and other aspects of the existing system.
b) A preliminary design is created for the new system.
c) A first prototype of the new system is constructed from the preliminary design. This is usually a scaled-down system, and represents an approximation of the characteristics of the final product.
A second prototype is evolved by a fourfold procedure:
a) Evaluating the first prototype in terms of its strengths, weakness, and risks.
b) Defining the requirements of the second prototype.
c) Planning an designing the second prototype.
d) Constructing and testing the second prototype.
At the customer option, the entire project can be aborted if the risk is deemed too great. Risk factors might involve development cost overruns, operating-cost miscalculation, or any other factor that could, in the customer's judgment, result in a less-than-satisfactory final product. The existing prototype is evaluated in the same manner as was the previous prototype, and if necessary, another prototype is developed from it according to the fourfold procedure outlined above. The preceding steps are iterated until the customer is satisfied that the refined prototype represents the final product desired.

The following diagram shows how a spiral model



4, SYSTEM REQUIREMENTS SPECIFICATION:

A requirement specification for a software system is a complete description of the behavior of a system to be developed. It includes a set of use cases that describe all the interactions the users will have with server. Use cases are also know as a functional requirements. Non-functional requirements are requirements which impose constraints on the design or implementation (such as performance engineering requirements, quality standards, or design constraints).

4.1 Functional Requirements:

Functional requirements specify which output file should be produced from the given file they describe the relationship between the input and output of the system, for each functional requirement a detailed description of all data inputs and their source and the range of valid inputs must be specified.

a) Pre Processing Module:

This is the first module. This module is to convert the input data to packets. And systems have to do the packet enhancements. Packets get from the data is extracted into packets.

b) Segmentation:

Data can be divided into packets. For switching system have to switch the data through multipath systems that balance the data.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

c) Action Recognition:

The Switching systems have to fetch the data to destination systems. This module is to find the human action with the use of parallel packet switches.

4.2 Non-Functional Requirements:

• Describe user-visible aspects of the system that are not directly related with the functional behavior of the system.

• Non-functional requirements include quantitative constraints, such as response time (i.e. how fast the system reacts to user commands.) or accuracy (i.e. how precise are the system numerical answers).

5. SYSTEM DESIGN:

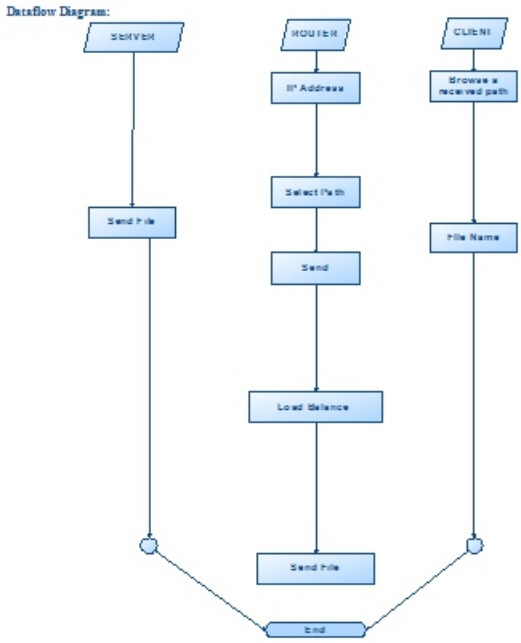
5.1 Data Flow Diagram:

The Data-flow diagram is a graphical representation of the "flow" of data through an information system. It differs from the flowchart. Flowchart as it shows the data flow instead of the control flow of the program. A data-flow diagram can also be used for the visualization of Data processing.

The system designer makes "a context level DFD" or, which shows the "interaction" (data flows) between "the system" (represented by one process) and "the system environment" (represented by terminators). The system is "decomposed in lower-level DFD" into a set of "processes, data stores, and the data flows between these processes and data stores." Each process is then decomposed into an even-lower-level diagram containing its sub processes.

Data flow diagrams ("bubble charts") are directed graphs in which the nodes specify processing activities and the arcs specify data items transmitted between processing nodes. A data flow diagram might represent data flow between individual statements or blocks of statements in a routine, data flow between sequential routines, data flow between concurrent process, or data flow in a distributed computing system, where each node represents a geographically remote processing unit.

Data Flow Diagram:



5.2 Unified Modeling Language:

The Unified Modeling Language (UML) is a standard language for writing software blueprints. The UML may be used to visualize, specify, construct, and document the artifacts of a software-intensive system. The UML is an appropriate for modeling systems ranging from enterprise information systems to distributed Web-based applications and even to hard real time embedded systems. It is a very expressive language, addressing all the views needed to develop and then deploy such systems. The UML is only a language and so is just one part of a software development method. The UML is process independent, although optimally it should be used in a process that is use case driven, architecture-centric, iterative, and incremental.

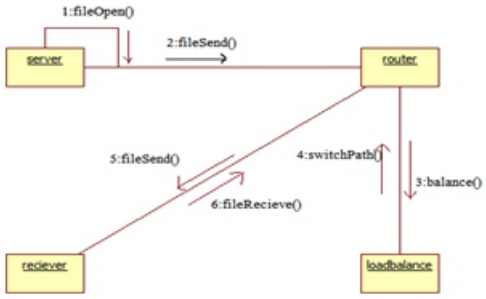
Use case diagram for system accesses.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

5.3 Collaboration Diagram:

A sequence diagram is dynamic and more importantly, is time ordered. A collaboration diagram is very similar to a sequence diagram in the purpose it achieves; in other words.

It shows the dynamic interaction of the objects in a system. A distinguishing feature of a collaboration diagram is that it shows the objects and their association with other objects in the system apart from how they interact with each other. The association between objects is not represented in a sequence diagram.



Collaboration diagram

6. SYSTEM IMPLEMENTATION:

6.1 MODULES:

1. Load-Balancing Scheme,
2. Multipath Switching System,
3. Multistage Multiplane Clos Switches,
4. Flow Slice.

a) Load-Balancing Scheme:

Interflow packet order is natively preserved besetting slicing threshold to the delay upper bound at .Any two packets in the same flow 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 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.

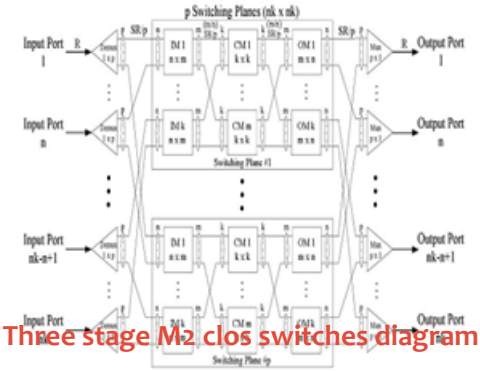
This table size depends only on system line rate and will stay unchanged even if scales to more than thousand external ports, thus guarantees system scalability.

b) 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 by. 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.

C) Multistage Multiplane Clos Switches:

The Multistage Multi plane Close network is a based switch by Chao et al. 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 close are composed of input demultiplexers and output multiplexers, respectively, having similar internal structures as those in PPS. Stages 2-4 of M2Clos are constructed by parallel switching planes; however, each plane is no longer formed by a basic switch, but by a three-stage Close Network to support large port count. Inside each Close 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 Close networks.



Three stage M2 clos switches diagram

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

D) Flow Slice:

A flow slice is a sequence of packets in a flow, where every intraflow interval between two consecutive packets is smaller than or equal to a slicing threshold Flow slices can be seen as miniflows created by cutting off every intraflow interval larger than . We depict the Cumulative Distribution Functions (C.D.F) of intraflow intervals in our traces. Most of the traces have more than 50 percent of their intervals larger than 1 ms, and more than 40 percent are larger than 4 ms. Two exceptions are PSC and FRG. The former is under a light load (≈10%); the latter is collected from a low-speed OC-12c network edge link. Thus, their weights are minimized. We denote the probability for an intraflow interval to be larger than by PC and the average packet count in the original flow by Co. After slicing, the average packet count in flow slices. Setting = 1ms, which leads to PC > 0:5, we obtain FC < 2. That is, each flow slice has no more than two packets in average. Even under a modest setting of = 4ms, we still have FC < 2:5. This reveals the very close load-balancing granularity of FS to the optimal packet-based solution.

CONCLUSION & FUTURE WORK:

A novel load-balancing scheme is proposed, namely, Flow Slice, based on the fact that the intraflow packet interval is often, say in 40-50 percent, larger than the delay upper bound at MPS. Due to three positive properties of flow slice, our scheme achieves good load-balancing uniformity with little hardware overhead and O(1) timing complexity. By calculating delay bounds at three popular MPS, system show that when the slicing threshold is set to the smallest admissible value at 1-4 ms, the FS scheme can achieve optimal performance while keeping the intraflow packet out-of-order probability negligible (below 10^6), given an internal speedup up to two.

The results are also validated through trace-driven prototype simulations under highly bursty traffic patterns. FS scheme had proven to be effective under strictly admissible traffic, but it is also important to know how it behaves under extreme traffic. The simulations with bursty real traces shed some light on this issue and suggest that it still work well. Actually, if only the slicing threshold is larger than the delay variation bound at all switching paths, packet order will be undisturbed.

Under bursty input traffic, the delay at all switching paths may increase synchronously, leaving its delay variation bound nearly unchanged.For future work, one of future works will be studying FS performance under Quality of Services (QoS) conditions, The FS scheme is validated in switches without class-based queues, As Quality of Services (QoS) provisioning is also critical in switch designs.

REFERENCES:

1] J. Bennet, C. Partidge, and N. Shectman, "Packet Reordering Is Not Pathological Network Behavior,"IEEE/ACM Trans. Networking, vol. 7,no. 6, pp789-798, Dec. 1999.

2] Z. Cao, Z. Wang, and E. Zegura, "Performance of Hashing-Based Schemes for Internet Load Balancing," Proc. IEEE INFOCOM, pp. 332-341, 2000.

3]S. Iyer, A. Awadallah, and N. McKeown, "Analysis of a Packet Switch with Memories Running Slower than the Line Rate," Proc. IEEE INFOCOM, pp. 529-537, 2000.

4] A. Aslam and K. Christensen, "Parallel Packet Switching Using Multiplexors with Virtual Input Queues," Proc. Ann. IEEE Conf. Local Computer Networks (LCN), pp. 270-277, 2002.

5] N. Brownlee and K. Claffy, "Understanding Internet Traffic Streams: Dragonflies and Tortoises," IEEE Comm. Magazine, vol. 40, no. 10, pp. 110-117, Oct. 2002.

6] I. Keslassy and N. Mckeown, "Maintaining Packet Order in Two- Stage Switches," Proc. IEEE INFOCOM, pp. 1032-1041, 2002.

7] L. Shi, W. Li, B. Liu, and X. Wang, "Flow Mapping in the Load Balancing Parallel Packet Switches," Proc. IEEE Workshop High Performance Switching and Routing (HPSR), pp. 254-258, 2005.

8] W. Shi and M.H. MacGregor, "Load Balancing for Parallel Forwarding," IEEE/ACM Trans. Networking, vol. 13, no. 4,pp. 790-801, Aug. 2005.

8] W. Shi, M.H. MacGregor, and P. Gburzynski, "A Scalable Load Balancer for Forwarding Internet Traffic: Exploiting Flow-Level Burstiness," Proc. Symp. Architecture for Networking and Comm.Systems (ANCS), 2005.

9] W. Shi and L. Kencl, "Sequence-Preserving Adaptive Load Balancers," Proc. ACM/IEEE Symp. Architecture for Networking and Comm. Systems (ANCS), 2006.