A New Lookup Model for Multiple Flow Tables of OpenFlow with Implementation and Optimization Considerations

Zhi Chen[1,2], Yulei Wu[1], Jingguo Ge[1], Yuepeng E[1]
[1]Computer Network Information Center, Chinese Academy of Sciences, Beijing, 100190, China
[2]University of Chinese Academy of Sciences, Beijing, 100049, China
Email: {chenzhi, wuyulei, gejingguo, eyp}@cstnet.cn

***Abstract*- OpenFlow has become the key standard for the southbound interface of software defined networking. The single flow table of OpenFlow implementation can lead to fast storage space growth, and finally cause table-overflow; the multiple flow tables can address this problem and provide greater efficiency and flexibility. Through analyzing the potential deployment challenges of OpenFlow, this paper proposes a new lookup model with implementation and optimization considerations for multiple flow tables in an OpenFlow switch. With the developed lookup model, the original single flow table is split into multiple sub-flow tables, and the fields in each sub-flow table are further divided into several categories according to different field types. Preliminary simulation results demonstrate that the proposed solution can effectively reduce the storage space of flow tables.**

***Keywords*- OpenFlow, lookup model, flow table, summary table, performance analysis**

I. INTRODUCTION

OpenFlow has become the key standard for the southbound interface of software defined networking (SDN) [1, 2, 3]. It decouples the control and data forwarding plane, and has been widely used for the low-cost configuration and optimization of traffic flows in Data-center and Campus networks to facilitate the traffic engineering and improve network performance.

In OpenFlow networks, the forwarding strategy and bandwidth assignment are determined in the per-flow basis, and, thus the design of flow table is one of the dominant issues [5]. Since OpenFlow specification v1.1, multi-stage flow table processing has been proposed to achieve an efficient and flexible table lookup; however, the implementation of such a proposal has not been given, even in the latest specification v1.3. The packets of traffic flow are processed against the flow entry in the flow table where a flow entry is mainly identified by the match fields and priority. Moreover, the structure of match fields is becoming more complex in the presence of new Internet architecture, new application type, and new media format. Thus, the single flow table of OpenFlow implementation can lead to fast storage space growth, and finally cause table-overflow; the multiple flow tables can address this problem and provide greater efficiency and flexibility [3, 4].

Due to the success achieved in Data-center and Campus networks [1, 2], the OpenFlow/SDN is demanding the deployment in large-scale environments (e.g., wide-area networks) to provide easy traffic engineering including bandwidth sharing and multipath transmission, high quality-of-experience (QoE), and the opportunities for researchers to evaluate their innovative designs without any interference to the practical network operation. However, the large-scale deployment of OpenFlow technologies possesses many challenges, due to the explosive growth of network bandwidth, the development of new network architectures, the increasing formats of new media and the diversification of network applications. Moreover, these factors have significant impact on the size, the structure and the lookup efficiency of flow table in OpenFlow: 1) the size of flow table will grow explosively in the future; 2) the number of fields will be increasing dramatically, the type of fields will change frequently, the structure of flow table will become more complex, and the matching of flow entries will be more costly with the mixture of diverse field types; 3) the increase in the processing speed of switch port requires an even higher demand for the efficiency of flow table lookup.

In our previous work [6], we defined the coexistence and conflict relationships between the fields by analyzing the match fields in OpenFlow specification v1.3, and proposed an algorithm H-SOFT to optimize the storage space of flow table. The proposed algorithm degrades the complex fields in a single flow table into multiple sub-flow tables with sub-set of original fields. The storage space compression of original flow table is achieved by assigning the fields of each flow entry into multiple sub-flow tables.

This paper further considers the storage space compression of each sub-flow table and the implementation proposal for the developed flow table lookup model. Specifically, the fields in each sub-flow table are divided into four categories: linear match field, exact match field, longest prefix match field, and range match field. A summary table is then adopted to make sure the consistency and correctness of the matching results. We further develop a feasible approach to optimize the summary table and solve its storage space explosion problem to be raised

in the practical implementation. The preliminary simulation results show that the proposed solution can effectively reduce the storage space of multi-flow tables with optimized summary table by around 50% under varying number of flow entries, in comparison with that using original summary table.

The rest of this paper is organized as follow. Section II shows the related work on flow table lookup. Section III presents the proposed lookup model for flow table of OpenFlow, and the optimization for its storage space. The feasibility of the proposed lookup model and the effectiveness of the presented optimization approach are validated through simulation experiments in Section IV. Finally, Section V concludes this study.

II. RELATED WORK

From the perspective of forwarding theory in computer networks, existing forwarding tables, such as forwarding information base (FIB) and access control list (ACL), can be treated as the subset of the flow table [7]. In OpenFlow architecture, the networking devices usually need to handle a large number of ACL-like rules. The compression of such rules is critical for the management and storage space optimization of networking devices [8].

Liu *et al.* [9] proposed an ACL Compressor to reduce the number of rules while maintaining the same semantics. However, the flow table of OpenFlow is different from the ACL; the actions in ACL are much simpler than those in flow tables of OpenFlow. Moreover, in the flow table, there may exist several entries against a single traffic flow for the purpose of different control granularities; the packets match flow entries in priority order.

It is difficult to scale the current TCAM-based or trie-based solutions for future routing tables due to the increasing table size and longer/varying prefix length, which demand for higher throughput [10]. The authors in [11] proposed the TCAM-based SPliT architecture where a *d*-dimensional classifier is split into k ($k \geq 2$) low dimensional classifiers, each of which is stored in its own small-size TCAM. A *d*-dimensional lookup is then equivalent to the k low dimensional and pipelined lookups with one lookup on each chip. TCAM is not the most appropriate solution for the lookup operation for all types of fields. In addition, due to power and cost reasons, TCAM is widely adopted for small-size routing tables [12, 13].

Since the specification v1.1, the OpenFlow has proposed multiple flow table processing to accelerate the lookup and compress the storage space. However, the overhead of actual flow table storage is related to the design of flow table structure which has not been given in the specification.

The lookup model for flow table of OpenFlow proposed in this paper can perform efficient table lookup because many match fields (i.e., linear match field, exact match field, etc.) of flow entries can be handled in parallel. Since multi-core processors have brought new opportunities to further improve the performance of flow-based traffic processing schemes [14], Li and Luo [15] extracted flow features with multi-core processors which can achieve 19.3% performance improvement. Using multi-core processors has been the current dominant trend to improve the performance of packet processing.

III. THE PROPOSED LOOKUP MODEL

The network programmability of OpenFlow is achieved through open and dynamic updating on the flow entries in flow table. The data packets are handled based on the actions of matched entries. The efficiency of flow table lookup significantly affects the performance of OpenFlow switches and the deployment of OpenFlow technology in large-scale networks. In this section, we propose a lookup model for flow tables to achieve efficient lookup and storage space compression of multiple flow tables in OpenFlow.

A. The Principle and Workflow of Lookup Model

In OpenFlow networks, each switch maintains a flow table with a set of flow entries; a logically centralized controller has the connections to all switches to add, update, and delete flow entries in flow tables based on the OpenFlow protocol. In previous work [6], we presented an optimized forwarding plane, where the single flow table is divided into k sub-flow tables using the H-SOFT algorithm, which is shown in the gray box of Fig. 1. In this paper, the proposed lookup model is based on the k sub-flow tables.

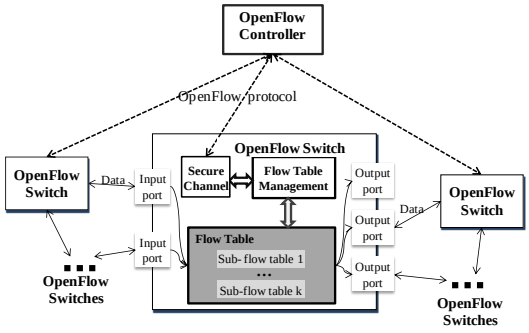

Fig. 1. The OpenFlow network with the flow table in the switch optimized by H-SOFT algorithm

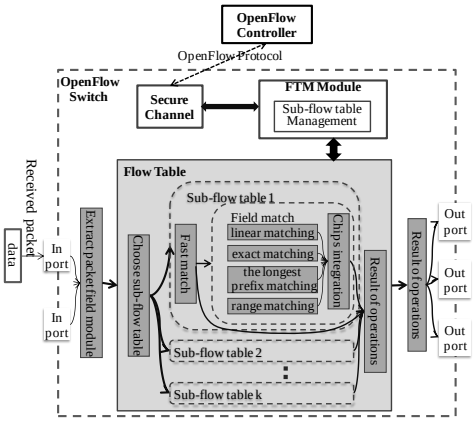

Fig. 2. The architecture of the proposed lookup model

As shown in Fig. 2, in the proposed lookup model, the flow table management (FTM) module is adopted to manage the flow tables. This module is independent in each switch and is transparent to the OpenFlow Controller. The controller can issue rules with the same format to different switches. The FTM module can adopt different management algorithms such as H-SOFT [6]. This module is scalable and can define the required and optional functions, where in this work, the sub-flow table management is required.

Recall that in Fig. 1, each of the n flow tables is divided into k sub-flow tables to optimize its storage space with the coexistence and conflict relationships between fields of flow entries [6]. The dimension of entries in sub-flow tables is lower than that of original flow table, and, thus the number of flow entries in each sub-flow table is smaller. That is because the entries in sub-flow tables only contain coexistence-relationship fields and the conflict-relationship fields cannot appear to occupy storage space.

Considering the sub-flow tables may contain various field types, they are further split into four categories: linear match field, exact match field, longest prefix match field, and range match field; the matching process of each field can be performed in parallel by different hardware implementation. We use the chip integration module (CIM) to handle the matching results of each field, which can be used to get the matching result of the sub-flow table.

The lookup operation of a received packet can be performed as follows: 1) the header fields in the arrival packet are extracted by the extract packet field module; 2) according to the extracted fields, one or multiple sub-flow tables are selected for lookup; 3) the fast match module in sub-flow table is checked first to get the result if matched; otherwise, go to Step 4; 4) different match fields are processed in parallel, and the results of each match field are collected to get the matching result of sub-flow table; 5) we compare the priority of matched entries in different sub-flow tables to obtain the final result; 6) According to the obtained result, the lookup operation ends.

B. The Implementation of Lookup Model

In the lookup model, the relationships between fields in an entry are broken when the sub-flow table is split. The values of each field are recoded in the corresponding matching module. Therefore, a summary table is required to ensure the consistency and correctness of the matching result for received packets, which should be semantically equivalent to that of original flow table and saved in the CIM.

The lookup model is implemented based on the hypothesis that the sub-flow tables contain at most two-dimension range match fields, linear match fields, and exact match fields, where the longest prefix match field can be treated as a special case of range match field.

Because multiplicative effect exists in the summary table and the number of entries will be explosively growing with the number of original entries increasing linearly. To solve this problem, we leverage a two-level summary table to compress the storage space of those fields whose values will be repeated when an entry is expanded to several entries in the summary table. Moreover, we only decompose one of the two range fields and convert the match modules of the two range fields to an equivalent tree structure, which can mitigate the multiplicative effects in the summary table.

In Fig. 3, an OpenFlow sub-flow table is given, as an example, to illustrate the implementation of lookup model.

Rules	Mac_Src	Mac_Dst	Proto	Port_Src	Port_Dst	
1	A1	B1	TCP	[1,10]	[20,100]	$\rightarrow e_1$
2	A1	B2	TCP	[3,5]	[30,50]	$\rightarrow e_2$
3	A2	B3	UCP	[6,8]	[80,81]	$\rightarrow e_3$
4	A3	B2	TCP	[3,8]	[60,65]	$\rightarrow e_3$

Fig. 3. An example of one sub-flow table

The match fields are split and their values are recorded in Fig. 4. For the two range fields, we decompose the values of Port_Src field for non-overlapping purpose, and the output of Port_Src field is the index for the Port_Dst field matching processing. The results due to the match fields splitting are shown in Fig. 4.

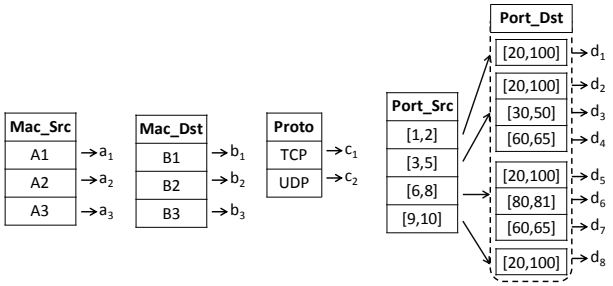

Fig. 4. The situation after match field dividing and recoding

Then, the matching results of each match module are collected by the CIM to produce the final matching result for the sub-flow table. In this paper, the proposed two-level summary table is shown in Fig. 5.

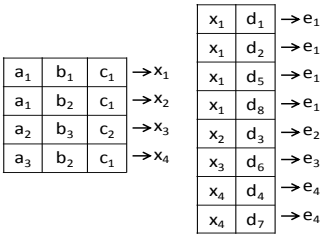

Fig. 5. The two-level summary table

From Fig. 5, we can find that the four entries in original sub-flow table (see Fig. 3) are expanded to eight entries in the summary table. If all match fields in sub-flow table are split, the number of entries in original summary table will increase significantly, which will be analyzed below.

For the field of Mac_Src, the value of A_1 exists in the entries of e_1 and e_2. With the field splitting, we need to

save the value of A_1 and record it as a_1; the other values, i.e., A_2 and A_3, of the Mac_Src field are recoded as a_2 and a_3, respectively. The fields of Mac_Dst and Proto are handled in the same manner. For the range match fields, e.g., Port_Src and Port_Dst, an equivalent two-layer tree structure is built. In the two-level summary table, the value (a_1, b_1, c_1) in the first-level table is mapping to x_1 which repeats four times in the second-level table. Through (x_1, d_1) in the second-level table, the final matching result e_1 can be obtained. The two-level summary table is used to reduce the storage space of summary table.

In Fig. 4, only the values of Port_Src field are decomposed for non-overlapping. The set of values in the field of Port_Src is denoted by r_1, and the number of values in r_1 is $||r_1||$. After the value decomposition, r_1 is converted to r_1', and, thus the values in r_1' are non-overlapping and the number of values in it is $||r_1'||$. Thus, we have $||r_1'|| \leqslant 2||r_1||-1$ [9]. Because a value in r_1 is an interval and it can be converted to two points, all the values in r_1 can be converted to $2||r_1||$ points in the case of maximization. Then, the maximization of $(2||r_1||-1)$ intervals can be obtained for non-overlapping with $2||r_1||$ points. Therefore, with such kind of decomposition, the growth in the number of values of range matching module is linear.

A value in r_1 may be expanded to $||r_1'||$ values in r_1'. The increase of entries in the summary table is, therefore, n^2 level when there is only one range match field in the sub-flow table and n is $||r_1||$. If the Port_Dst range field is also decomposed for non-overlapping, multiplicative effect will be significant and the increase of entries in summary table is n^3 level. In our proposed approach with two-layer tree structure, the increase of entries is n^2 level with two range fields in the sub-flow table.

IV. PRILIMINARY SIMULATION RESULTS AND ANALYSIS

To evaluate the effectiveness of the proposed approach for summary table optimization in the implementation of lookup model, we develop a software simulator based on Python to mimic the behavior of lookup model. All simulation results are collected from a personal computer with Windows 7 operating system, Intel Core (TM) 2, 2.66GHz CPU and 3.46GB memory space.

OpenFlow is currently in the process of development, it is still hard to collect the data of realistic flow tables. Therefore, in this paper, we convert the header of network packets into the corresponding entries of flow table. The real dataset of network packets is collected from the egress router of Guangzhou branch of China Science and Technology Network (CSTNet), an academic and research network in China. The entries in the flow table contain several match fields: IPv4_SRC, IPv4_DST, IPv6_SRC, IPv6_DST, IP_PROTO, TCP_SRC, TCP_DST, UDP_SRC, and UDP_DST.

Through preliminary simulation results with different scale of sub-flow tables, we separately compute the storage space of sub-flow tables with original summary table, optimized summary table, and the initial sub-flow table; the results are shown in Fig. 6. From the figure, we can find that the storage space of sub-flow tables with original summary table is larger than that of initial sub-flow table, especially under a moderate to large number of entries in sub-flow tables; the storage space of sub-flow table with optimized summary table is smaller than that of initial sub-flow table. In addition, the growth of storage space on sub-flow table with optimized summary table is slower than that of sub-flow table with original summary table in terms of the increased number of entries in sub-flow table. The phenomenon emphasizes the effectiveness and correctness of the proposed solution.

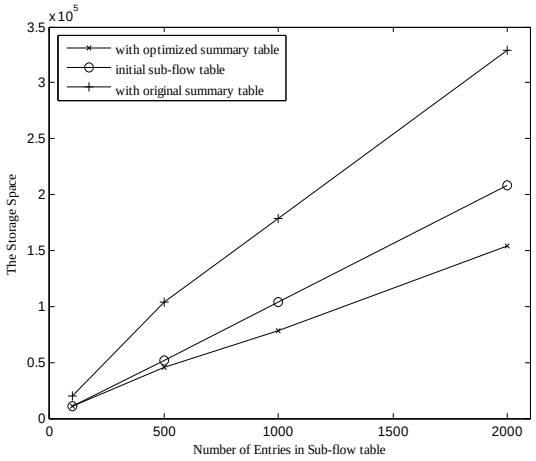

Fig. 6. The comparison of storage space under different conditions

Fig. 7 depicts the ratio of storage space between the sub-flow tables with optimized summary table and that with original summary table against the varying number of entries in the sub-flow table, i.e., 100, 500, 1000, and 2000 entries. The ratio is calculated based on: $R=(S_{opt}-S_{ori})/S_{ori}$, where S_{opt} and S_{ori} are the storage space of sub-flow tables with optimized summary table and original summary table, respectively.

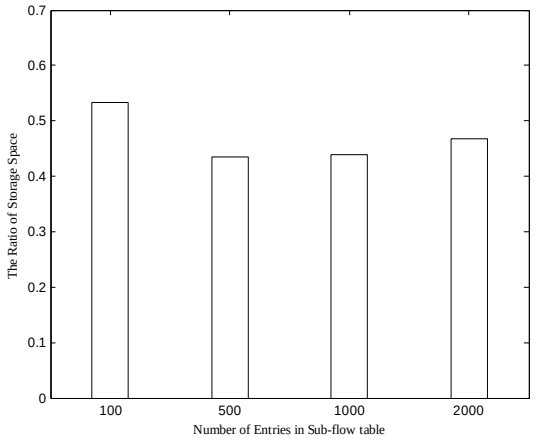

Fig. 7. The ratio of storage space between sub-flow table with two different summary tables

From Fig. 7, we can observe that the ratio is around 50% with the varying number of flow entries in sub-flow tables, which validates again the effectiveness and correctness of the proposed scheme.

In addition to the dataset collected from one egress router of CSTNet (Dataset A), we adopt another real dataset of firewall (Dataset B) and convert the rules into the corresponding entries of flow table. Fig. 8 shows the comparison of storage space for flow tables with optimized summary table and original summary table using Dataset A and Dataset B. The figure shows that the storage space with optimized summary table is always smaller under different datasets.

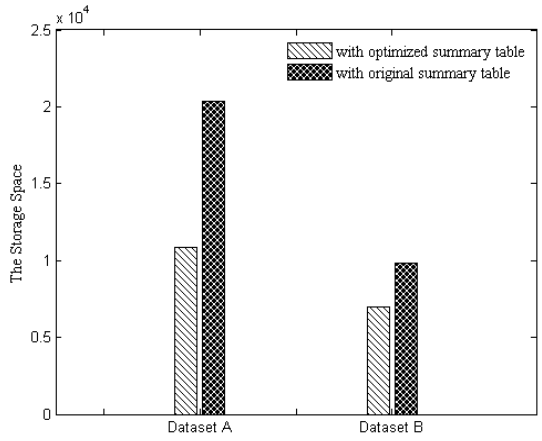

Fig. 8. The storage space comparison based on different real datasets

The results demonstrate that the effectiveness and correctness of the optimization approach for the storage space of summary table, which is important in the implementation of lookup model for multi-flow tables of OpenFlow.

V. CONCLUSIONS

This paper has proposed a new lookup model with implementation and optimization considerations for multi-flow tables of OpenFlow. In the lookup model, the flow table is split into multiple sub-flow tables based on the proposed H-SOFT. The fields in each sub-flow table are further divided into four categories: linear match field, exact match field, longest prefix match field, and range match field. A summary table in the model is used to make sure the consistency of the matching results. In the implementation of lookup model, we have proposed a feasible approach to handle the storage space explosion problem of summary table. Preliminary simulation experiments have shown that the proposed approach can effectively reduce the storage space of sub-flow tables with split fields. The compression rate of storage space between the sub-flow tables with original and optimized summary table is around 50% with varying number of flow entries. Moreover, the approach is also effective using different real datasets with diverse match fields.

ACKNOWLEDGEMENTS

This work is partially supported by the National Natural Science Foundation of China under Grant No. 61303241, National Program on Key Basic Research Project (973 Program) under Grant No. 2012CB315803, the Strategic Priority Research Program of the Chinese Academy of Sciences under Grant No. XDA06010201 and the National High-Tech R&D Program of China (863 Program) under Grant No. 2013AA013501.

REFERENCES

[1] Vaughan-Nichols S J. “OpenFlow: The Next Generation of the Network?” *Computer*, 2011, 44(8), pp. 13-15.

[2] Mckeown N, Anderson T, Balakrishnan H, Parulkar G, Peterson L, Rexford J, Shenker S, and Turner J. “OpenFlow: Enabling Innovation in Campus Networks,” *ACM SIGCOMM Computer Communication Review*, 2008, 38(2), pp. 69-74.

[3] OpenFlow 1.3.0 Specification. https://www.opennetwork-ing.org/images/stories/down-loads/specification/openflow-spec-v1.3.0.pdf [25 June 2012].

[4] Natarajan S, Ramaiah A, Mathen M. “A Software Defined Cloud-Gateway Automation System Using OpenFlow,” in *Proc. of 2013 IEEE International Conference on Cloud Networking (CloudNet)*, 2013, pp. 219-226.

[5] Matsumoto N and Hayashi M. “Performance Improvement of Flow Switching with Automatic Maintenance of Hash Table Assisted by Wildcard Flow Entries,” in *Proc. of 2012 10th International Conference on Optical Internet (COIN)*, 2012, pp. 12-13.

[6] Ge J, Chen Z, Wu Y, and E Y. "H-SOFT: A Heuristic Storage space Optimisation Algorithm for Flow Table of OpenFlow," *Concurrency and Computation: Practice and Experience*, 2014, DOI: 10.1002/cpe.3206

[7] Feng T, Bi J, and Hu H. “OpenRouter: OpenFlow Extension and Implementation Based on A Commercial Router,” in *Proc. of 2011 19th IEEE International Conference on Network Protocols (ICNP)*, 2011, pp. 141-142.

[8] Daly J, Liu A X, and Torng E. “A Difference Resolution Approach to Compressing Access Control Lists,” in *Proc. of 32nd Annual IEEE International Conference on Computer Communications (INFOCOM)*, 2013, pp. 2040-2048.

[9] Liu A X, Torng E, and Meiners CR. “Compressing Network Access Control Lists.” *IEEE Transactions on Parallel and Distributed Systems*, 2011, 22(12), pp. 1969-1977.

[10] Huang Z, Lin D, Peir J K, et al. “Fast Routing Table Lookup Based on Deterministic Multi-hashing,” in *Proc. of 2010 18th IEEE International Conference on Network Protocols (ICNP)*, 2010, pp. 31-40.

[11] Meiners C R, Liu A X, Torng E, et al. “Split: Optimizing Space, Power, and Throughput for TCAM-based Classification,” in *Proc. of the 2011 ACM/IEEE Seventh Symposium on Architectures for Networking and Communications Systems*, 2011, pp. 200-210.

[12] Akhbarizadeh M J, Nourani M, Vijayasarathi D S, et al. “PCAM: A Ternary CAM Optimized for Longest Prefix Matching Tasks,” in *Proc. of Computer Design: VLSI in Computers and Processors*, 2004, pp. 6-11.

[13] Noda H, Inoue K, Kuroiwa M, et al. “A Cost-efficient High-performance Dynamic TCAM with Pipelined Hierarchical Searching and Shift Redundancy Architecture.” *IEEE Journal of Solid-State Circuits*, 2005, 40(1), pp. 245-253.

[14] Wang D, Xue Y, Dong Y. “Memory-efficient Hypercube Flow Table for Packet Processing on Multi-cores,” in *Proc. of Global Telecommunications Conference (GLOBECOM)*, 2011, pp. 1-6.

[15] Li S, Luo Y. “High Performance Flow Feature Extraction with Multi-core Processors,” in *Proc. of 2010 IEEE Fifth International Conference on Networking, Architecture and Storage (NAS)*, 2010, pp. 193-201.