International Journal of Grid and Distributed Computing
Vol. 9, No. 1 (2016), pp.181-192
http://dx.doi.org/10.14257/ijgdc.2016.9.1.18

An SDN Based CBT for Distributed Shared Memory†

Qiang Gao[1*], Weiqin Tong[1], Samina Kausar[1] and Shengan Zheng[2*]

[1]*School of Computer Engineering and Science Shanghai University*
Shanghai, China
[2]*Department of Computer Science and Engineering*
Shanghai Jiao Tong University
Shanghai, China
[1]*redwolf19867@126.com*, [1]*wqtong@shu.edu.cn*, [1]*saminamalik7@yahoo.com*,
[2]*venero1209@sjtu.edu.cn*

Abstract

It is a challenging task to provide an easy–to-use programming interface for distributed shared memory in parallel computing that demands high availability as well as effective synchronization. The current implementations typically use conventional Ethernet as communication medium using broadcast or multicast for synchronization mechanisms. In general, multicast is more efficient than broadcast, as it uses multicast tree to transfer data instead of multicast flooding. Multicast avoids redundant transmissions and lessens network contention. These properties make it significant for distributed shared memory. However, such solution has limited control on response time and process scheduling. This causes uncertainty of time spent in synchronization and interruption of unrelated processes in the same multicast group. In this research, we propose a dynamic SDN multicast mechanism to improve efficiency of synchronization for distributed shared memory by using an SDN-based multicast mechanism and a dynamic core based tree algorithm. The forwarding rule reduces the number of consistency-related messages without interfering the unrelated processes in the same multicast group. The dynamic core-based tree algorithm builds a multicast tree for every multicast group, hereby, SDN controller dynamically adjusts multicast tree according to joining or leaving of nodes. The experiments on the prototype system demonstrate that it could provide better performance compared to the conventional Ethernet based solution in terms of response time and process scheduling in the homogeneous testing environment.

Keywords: *multicast; SDN; core based tree; openflow; steiner tree*

1. Introduction

The motivation in implementing the distributed shared memory (DSM) is to provide a shared-memory based programming model for the distributed application design. In the big-data era, to analyze massive amounts of data, needs large-scale clusters and highly efficient parallel programming models. Especially in-memory processing systems for real-time big data analysis, demand large quantities of bulk data to reside and transmit in memory for efficient processing [1][2]. DSM is naturally suitable to be used as a programming model for this scenario.

†Supported by Innovation Action Plan supported by Science and Technology Commission of Shanghai Municipality (15DZ1100305).
Qiang Gao is the corresponding author.
This paper is a revised and expanded version of a paper entitled 'A design and implementation of multicast for SDN based DSM' presented at 9th International Conference on Future Generation Communication and Networking, Jeju Island, Korea, November 25 - 28, 2015.

ISSN: 2005-4262 IJGDC

Classical Ethernet architecture seems to be the natural carrier to be used as an interconnection mechanism in DSM. Recently, many implementations typically use conventional Ethernet as communication medium using broadcast or multicast for synchronization mechanisms. Unfortunately, the DSM leveraging broadcast or multicast on conventional Ethernet cannot satisfy the requirements of real-time applications. This is particularly true for two major reasons, one is the uncertainty of data forwarding and the other is interference of unrelated processes in the same multicast group. Consequently, performance degradation occurs in DSM system. A classic Ethernet-based DSM is often limited by its interconnect performance. From the above-mentioned reasons, the current DSM synchronization does not hold true for real-time applications.

Software-Defined Networking (SDN) is an emerging architecture that is dynamic, manageable, cost-effective, and adaptable, making it ideal for the high-bandwidth, dynamic nature of today's applications. This architecture decouples the network control and forwarding functions enabling the network control to become directly programmable and the underlying infrastructure to be abstracted for applications and network services [3]. In SDN architecture, switches are only responsible for data forwarding and SDN controller is in charge of delivering a set of rules instructing how to deal with network flows to switches through a network control interface .A significant feature of SDN is that it allows user applications to have a whole network overview and interaction with SDN controllers, applications can easily reflect its function to forwarding rule.

As we discussed above, classical Ethernet architecture based DSM is often limited by its interconnect performance in real-time applications. Considering the before mentioned properties of SDN, DSM and other high performance computing software have good chances to improve performance by means of SDN[4]. In this research, we propose an SDN-based multicast for distributed shared memory to speed up distributed applications by reducing network contention delays and node computation overhead. We present a dynamic SDN multicast mechanism to improve efficiency of synchronization for distributed shared memory. In addition to these, we design a dynamic core based tree algorithm that helps SDN controller to dynamically adjust multicast tree and build time-predictable routes for the synchronization.

2. Related Work

[5] [6] offered a prototype system Brazos and presented a method to explicitly reduce the amount of consistent traffic on the conventional Ethernet network, using multicast to minimize consistency-related communication. All the processes having same share memory are added to one multicast group. Once a process is requested to change its new data to distributed share memory, the synchronous messages will only be sent to the processes in one multicast group along the multicast tree. In line with this philosophy, there will be only one synchronous message on each edge. The processes which do not have same share memory will not be interrupted by extraneous messages. [5] also introduced two issues caused by multicast, one is useless multicast traffic and the other is multicast conflict. Former arises by continuous multicast update messages in distributed share memory which is no longer in use. Latter occurs when a process receives an unexpected update for certain distributed share memory while this process is waiting for an update. Brazos used a dynamic copyset reduction algorithm and an early update mechanism to alleviate these issues.

[7] proposed a real-time operating system with distributed shared memory. The distributed shared memory uses a real-time network called FlexRay, which is based on a Time Division Multiple Access protocol. The consistency in DSM is maintained according to the order of data transfer through FlexRay without using inter-node synchronization. The worst case response time of the DSM is predictable by the well configuration of FlexRay communication. The distributed shared memory relies on

proprietary protocol and the support of real-time operating system. It is not a generic solution for the most application scenarios.

[8]presented a new approach for in-memory distributed storage called RAMCloud, where data is kept entirely in DRAM and the system is created by integrating the DRAM of thousands of servers. The performance of RAMCloud is 100~1000 times better than the best current solution. Such performance is achieved through specialized networks such as Infiniband. But most existing datacenters use classical Ethernet as underlying network, with typical round-trip times in the range of 300~500 μs. The network cannot meet the requirements of RAMCloud. It is thus clear that the greatest obstacle of RAMCloud to be used in datacenters is latency in the network switches.

[9] presented a new tree-based multicast routing protocol which uses an advanced label mechanism to reduce the required control packets for topology initialization and path construction. Meanwhile, a new mechanism is introduced to select a better routing path for transmission. However, the advantage of this mechanism is supported by particular protocol and cannot be used for Ethernet.

[10] [11] tried to use SDN as communication medium to transfer video data. These prototype systems run as SDN application with the help of north bound interface of SDN controller. As SDN has a whole network overview and adaptable quality, the video transmission can be set to optimize route according to time constraint, bandwidth requirement and network congestion. Under the guidance of this tactics, video transmission can be achieved in a predictable time. [12] [13] [14] studied how to use SDN to realize IP multicast and show how many benefits can be grasped from SDN based multicast. Though these SDN-based multicasts are manageable, cost-effective and predictable yet all of them are only suited for one-point-to-multiple-point scenario. But in distributed shared memory application scenario, every node using distributed shared memory is not only a source but also a receiver; it is a multiple-point-to-multiple-point scenario. The above multicast tree algorithm is not fit to this situation.

Our earlier work [15] [16] presents an SDN-based distributed share memory which has some limitations as compared to our current work of improving the transmission performance. Moreover, that work does not address the concept of core-based tree.

3. Design and Implementation

Our SDN multicast can be divided into two parts, one is registered and forward mechanism of multicast and the other is dynamic core based tree algorithm. The former mainly collects topology information and manages multicast group. The latter adapts multicast tree dynamically for every multicast group.

3.1.Implemenatation of Distributed Shared Memory

Our DSM is implemented in user space and use virtual memory system to provide a level of consistency checking in hardware, employing system calls to protect shared memory regions of programs. Not only that, the distributed share memory is managed and operated by group, every process quote a SDN-DSM will be added to the group and binding with a certain DSM ID. This implementation can be beneficial for user to flexibly define their own DSM granularity. *Table. 1* shows the definition of SDN-DSM object structure. Out of this, we give the highest priority to SDN-DSM-related data to ensure DSM get fastest response time in underlying network.

Table 1. DSM STRUCTURE

Field	Bits	Notes
DSM_ID	32[a]	ID of this SDN-DSM object
DSM_size	32	Memory size of this SDN-DSM object
process_count	32	Number of processes attached to this DSM object
multicast_address	48	Multicast address for this DSM object

Different from the SDN multicast implemented in [17], the SDN multicast proposed in this paper is process-oriented. It means that, all the processes using same distributed shared memory will register to same multicast group. Every node which registered to certain multicast group can be added to or removed from the group during runtime. The nodes which are no longer actively using the DSM will not be continued by multicast, while the other nodes constantly send DSM-updates to same multicast group.

Registered mechanism consists of two parts: multicast registry client and multicast registry server. Multicast registry client is deployed on the nodes which need to bind to multicast group. Multicast registry server is deployed on physical server with SDN controller as an SDN application. The former is responsible for registering local process to multicast registry server and mapping DSM space with local space. The latter is responsible for managing mapping relationships between DSM object and process. Multicast registry server invokes a function of altering multicast tree through SDN controller.

The core of multicast registry server is a multicast group update mechanism, an array of list pointers and the corresponding lists. The array is DSM ID based and every element points to a multicast group list. Multicast group list saves information of process that bound to a certain DSM ID. Each node of list contains an IP address that belongs to the node using certain DSM. When a DSM packet arrives in OS, it will trigger a socket event. The Socket event will generate a system signal. Subsequently, the signal-handling function will update array and lists for multicast registry server. The following *Table. 2* shows how the multicast group updating mechanism modify multicast forwarding rule through SDN controller.

Table 2. MULTICAST GROUP UPDATING MECHANISM

```
1: Getting message in listing socket;
2: Extracting action tag, DSM_ID and IP address from message;
3: if( action tag == binding )
4:   if( (DSM_ID >= 0 ) && (DSM_ID <= m) )
5:     head = updates_array[DSM_ID];
6:     while(head)
7:       if( head->ip_addr == IP address )
8:         return ok;
9:       end if
10:      head = head->next;
11:    end while
12:    head = updates_array[DSM_ID];
13:    p = Allocates space for new node of list;
14:    while(head !=NULL )
15:      head = head->next ;
16:    end while
17:    head->next=p;
18:    p->next=NULL;
```



```
19:     Call SDN controller to update multicast forwarding rule;
20: end if
21:    else
22:       return error;
23:    end else
24:end if
25: if( action tag == unbinding )
26:    if( (DSM_ID >= 0 ) && (DSM_ID <= m) )
27:        p1 = updates_array[DSM_ID];
28:        while (p1-> ip_addr == IP address && p1->next != NULL)
29:            p2 = p1;
30:            p1 = p1->next;
31:         end while
32:    if ( p1-> ip_addr == IP address)
33:       if (p1 == updates_array[DSM_ID];)
34:          updates_array[DSM_ID] = p1->next;
35:       end if
36:       else
37:          p2->next = p1->next;
38:       end else
39:       Free node space in memroy;
40:       p1 = NULL;
41:       Call SDN controller to update multicast forwarding rule;
42:    end if
43:    else
44:       return error;
45:    end else
46:   end if
47:end if
```

We encapsulate the functions of multicast registry client to two API functions, programmer can use binding_DSM(int DSM_ID, char *local_memory_addr) and unbinding_DSM(int DSM_ID) to binding and unbinding process to DSM. DSM_ID is the parameter for indicating which group to join, and local_memory_addr indicates the local memory space to mapping with certain DSM. After process calls binding_DSM(int DSM_ID, char *local_memory_addr), multicast registry client will send message to multicast registry server to register local process to multicast registry server. If multicast registry server accepts this register, it will call SDN controller to install forwarding rule for the process. Then multicast registry client creating a DSM socket to synchronous data to local memory. On the contrary, once process call unbinding_DSM(int DSM_ID), multicast registry client will send message to multicast registry server to unregister local process from multicast registry server. Multicast registry server will remove the process from registry list, and multicast registry client will close DSM socket. The registered and forward mechanism takes place as follows:

1. Share-memory-producer process sends notification to multicast order server to claim it will multicast a share memory data to multicast group. The notification covers source IP address of node, DSM objects ID and trigger time.
2. Multicast order server collects notifications in signal driving mode, extracts source IP address, DSM ID and NTP timestamp from the notification.
3. Multicast order server sends request to SDN controller to get topologic of underlying network. Then update multicast tree according to the topologic information.

4. Multicast order server calls SDN controller to install forwarding rules in SDN switches through Openflow protocol.
5. Multicast order server sends response to share-memory-producer process. It allows the process to broadcast share memory data to multicast group.
6. Share-memory-producer process broadcasts its share memory data to multicast group.

Figure. 1 shows the procedures described above.

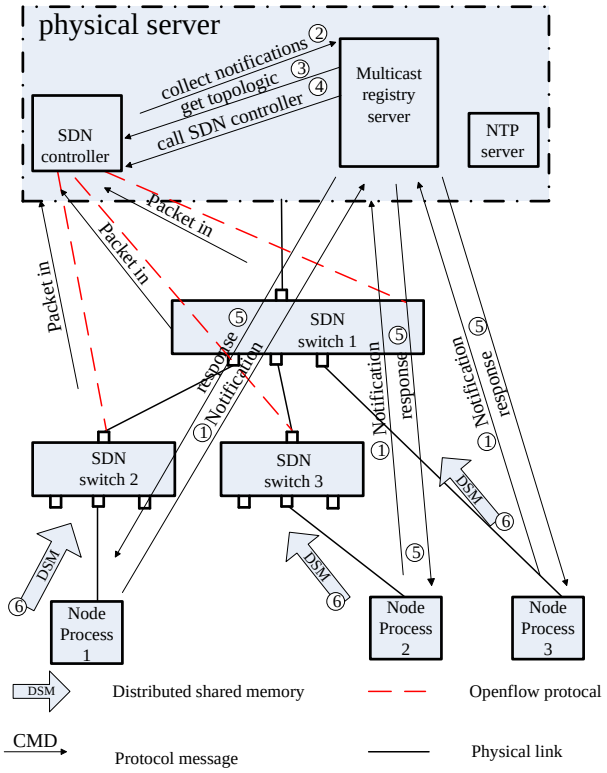


Figure 1. Registered and forward Mechanism

3.2. Dynamic core based Tree Algorithm

Multicast trees can be categorized into two types, one is source tree and the other is shared tree. Source tree multicast mechanism builds a multicast tree for every source and the tree connects the source and other nodes with the shortest path, as shown in the part A of Figure 2. It shows good performance on end-to-end communication among source and other nodes. But this multicast mechanism needs to keep spanning tree for every source. It means that network has to maintain a spanning tree according to self-state. This mechanism requires a significant amount of computing resources. However, in distributed shared memory application scenarios, network topology changes dynamically with entry and exit of nodes. In this case, computing cost is not acceptable. Shared tree multicast mechanism builds a multicast tree for the whole multicast group. It cannot provide an optimum solution for each path between source and the other nodes. But the network only keeps one spanning tree for the whole multicast group consuming less computing resources. This feature makes it suitable for distributed shared memory application scenarios.

PIM(Protocol Independent Multicast) and CBT(Core Based Tree) are the most popular solutions for building shared tree[18]. PIM can build a one-way multicast tree and the data is transmitted from the source to the other nodes in only one direction. It is not a suitable way to handle distributed shared memory problems. CBT takes one forwarding

device as a root to build a shared tree for all the nodes. When the source sends data to the other nodes, the data is first transferred to the root and then forwarded to the other nodes by the root, as shown in the part B of Figure. 2.

In distributed shared memory system, any node at any instant may propagates updates to multicast group or receives updates from multicast group. It means that, the synchronization of distributed shared memory is a multi-source multicast scenario. The core issue of multi-source multicast is multicast routing problem, i. e. how to construct the least cost tree from source to all the destination nodes. This problem can be formed into a steiner tree problem in graph theory. But Steiner tree has been proved to be NP-complete problem so solution cannot be found in polynomial time. To get an optimal approximation solution, we need to find a heuristic algorithm for SDN based multicast.

In distributed shared memory application scenario, every node using distributed shared memory is not only a source but also a receiver. If we use algorithms like Dijkstra to maintain the least cost tree for every DSM in every node, it will consume a lot of computing resources. The registered and forward mechanism for DSM allows nodes to join or leave multicast group dynamically. It means that, SDN controller has to reconstruct multicast tree for each change. This situation further aggravates the consumption of computing resources. In the meantime, the response time of distributed shared memory will increase.

To address this issue, we try to use core based tree (CBT) as multicast tree for synchronization of DSM. Core based tree algorithm will construct a shared multicast tree for every multicast group. All the nodes in one multicast group share the same multicast tree, but not have their dedicated multicast tree. CBT is very applicable to multiple-point-to-multiple-point scenario due to its non-source-oriented feature. No matter which node send the multicast data, the SDN switch will forward the data to other node through the same tree. Core based tree covers that part of network which connect to all the nodes in one group. This property helps it to be well adjusted for large-scale group communication. Accordingly, DSM based on this multicast tree algorithm will get better extensibility.

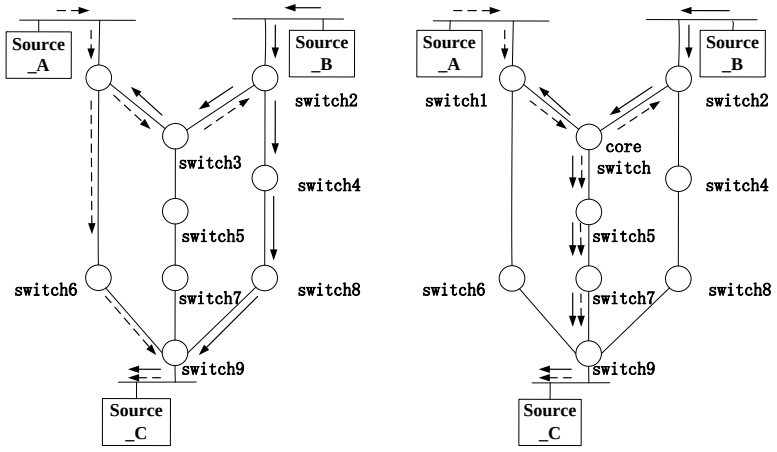


Figure 2. Independent path and shared path

In classical Ethernet architecture, a router should be selected as the core of multicast tree for CBT. Nodes which try to join to multicast tree will first announce themselves to edge router. When Edge router receives this announcement, it will send request message to core router. The request message will approach to core router hop-by-hop and record its route. When the request message arrives at the previous router of multicast tree, the transmission of request message will stop and the router will return license message to the edge router. Upon completion of these steps, nodes will get the route to CBT and send data to multicast group.

CBT algorithm in classical Ethernet architecture has many limitations. Firstly, it is very complex to add nodes in multicast tree. Secondly, it is very difficult to fix the suitable core router from classical Ethernet architecture. Thirdly, it is uncontrollable in managing multiple sources to inject data in CBT. Our implementation avoids these limitations completely though whole network overview by using SDN. Firstly, Multicast registry server gets topology information of whole network through SDN controller. Then finds the switches which have the shortest average distance with other switches. Secondly, CBT algorithm constructs an original CBT for every multicast group according to the registry information of multicast registry server. Thirdly, SDN controller installs forwarding rule in SDN switch following the CBT. Finally, every node attached to the core based tree will be treated as a leaf node. When nodes register to multicast group or unregister from the group, the multicast registry server will inoculate or prune nodes from multicast tree and refresh the forwarding rules of SDN switches. Once CBT has been constructed, inherently, core router is no longer different from other routers. Multicast data can be injected into multicast group from any switch of the CBT, switches will transfer data to other neighboring switches. SDN multicast therefore obtains shorter transmission delay and less transmission overhead. The basic idea of our CBT algorithm is described as below:

Since the root constitutes the only point of distribution of traffic to other vertices, its ideal location is in “the middle” of the tree which means that the core switch should be located in "the middle" of multicast group. In graph theory, this problem can be formed to find the center of graph. The center of graph has been defined as follows:

Definition 1 (Eccentricity). Let a digraph G(V,E) be given, v is a vertex of G. Then, the farthest path between v and other vertices $e_G(v) = \max\{d(u,v)\}$ is called the eccentricity of v.

Definition 2 (Radius). The radius of G is the shortest eccentricity for all vertices , the expression is $r(G) = \min\{e(v)\}$.

Definition 3 (Center). The Center of G is the set of vertices which have shortest r(G), defined as C(G).

For classical Ethernet architecture, choosing C(G) for multicast group is infeasible as the dynamic joining and leaving of nodes are unknown at group’s run time. But with the help of SDN controller, our CBT algorithm not only calculates the weight of edges, but also tries to keep core in “the middle” of the tree. It means that, it should find an SDN switch b that makes sources $n_1,n_2,n_3,...,n_n$ having a wider range of coverage. In the beginning of the algorithm, an original CBT will be constructed according to the network topology information. It uses Dijkstra algorithm to find the shortest path from switch b to all source nodes ($s_i \rightarrow b$) and the shortest path from b to C(G). Lastly, takes bandwidth w_i and traffic q_i as weight of edges for calculating total weight. The tree which has maximum weight will be the desired minimum cost CBT, and switch b will be the access point switch. The *Table. 3* shows how Dynamic core based tree algorithm finds CBT for underlying network.

Table 3. Pseudo-Code for CBT Algorithm

CBT ALGORITHM	
1:	Triggered by the change of multicast group;
2:	Get new topology of network from SDN controller;
3:	Probe C(G) of network.
4:	Construct the original CBT based on C(G);

5: **for** $\forall b \subset \{G - \{n_1, n_2, ..., n_n, C(G)\}\}$
6: Dijkstra($s_i \rightarrow b$);
7: Dijkstra($b \rightarrow C(G)$);
8: **end for**
9: **for** each edge $e \in T_{q_i}^{s_i \rightarrow b}$
10: set weight of edge $f(w_i - q_i)$;
11: **end for**
12: **for** each edge $e \in T_{q_1 \oplus q_2 \oplus ... \oplus q_n}^{b \rightarrow C(G)}$
13: set weight of edge $f(w_i - q_1 \oplus q_2 \oplus ... \oplus q_n)$;
14: **end for**
15: Calculate total weight;
16: Find max weight to get desired CBT;
17: Install forwarding rules to SDN switches according to CBT;

Our CBT algorithm is based on Dijkstra algorithm, therefore it can be solved in a certain amount of time and the time complexity is O (n^2). The algorithm will execute in N+1 times for N multicast sources. In addition, to find the minimum cost CBT has to perform comparisons in n times for n nodes. Consequently, total time complexity is O (Nn^3).

4. Experimental Design and Analysis

To indicate the transmission efficiency of our SDN based CBT, we compared transmission delay with broadcast and multicast source tree using the same network topology and same forwarding device. To ensure our SDN based CBT can be compared with broadcast and multicast source tree using the same underlying network, we build a classical switch network in SDN architecture for broadcast and multicast source tree. Our SDN based CBT allows data to be forwarded by the switches with same level of performance. The experiments are performed using mininet simulation environment **[19]**. In mininet, user can set bandwidth and network topology by themselves. To show transmission efficiency more clearly, we set the bandwidth to 1 Mb/s and send 1M byte data through the network. The network topology is created by the following steps:

1. Attaching all the nodes **by** a tree to make sure the connectivity of graphs.
2. The vertex has degree 1 randomly connected to other vertices.
3. Creating random network topology following waxman algorithm.
4. When average degree of the network reaches preset value, we get the final network topology .

In our experiments, we use 10 nodes and set average degree to 3. The results show that the multicast source tree has the best performance, as shown in the Figure. 3,as it builds a multicast tree for every source, furthermore, the tree connects source and other nodes with the shortest path. Broadcast has the worst performance and deteriorates more with the growing node count. The performance of SDN based core based tree is more poor than source tree, but much better than broadcast. Considering maintenance costs, SDN based core based tree is an optimal solution.

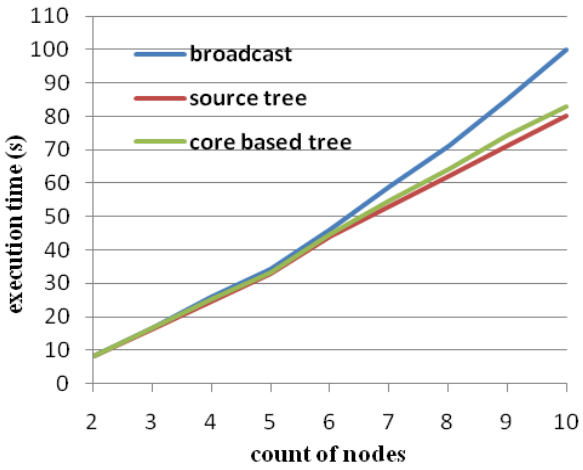


Figure 3. Performance Comparison of Three Different Communication Mechanisms

Pantou[20] is a software for OpenWrt[21], it can turn a commercial wireless router/Access Point to an OpenFlow-enabled switch. For better comparison of the performance of state-aware data transmission mechanism with common transmission mechanism, we form the testing environment based on 4 commercially manufactured routers. We burn OpenWrt system to these routers and install Pantou on it. In this case, experiments are performed on a cluster system which consists of 4 SDN-switches/commercially manufactured routers, one SDN controller and 4 nodes.

We use a simple producer/consumer example to evaluate the performance of process-oriented dynamic SDN multicast. One producer updates DSM data, and places it for worker process to consume and compute. Figure.4 shows a simple producer/consumer example. The acceleration effect of process-oriented dynamic SDN multicast was not obvious in a distributed application with few DSM data. But when the amount of DSM exceeds a certain number in a distributed application, the application can benefit a lot from process-oriented dynamic SDN multicast.

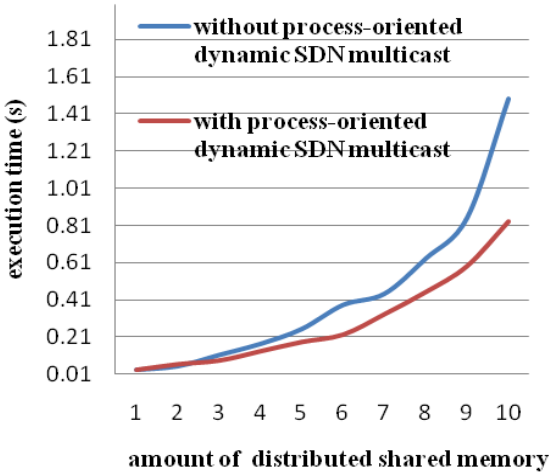


Figure 4. Performance Comparison between SDN based DSM and Classical Network based DSM

5. Conclusion

In this paper, we presented an experimental study showing how SDN multicast can be used to optimize synchronization of DSM. We propose a dynamic SDN multicast mechanism to improve efficiency of synchronization for distributed shared memory by

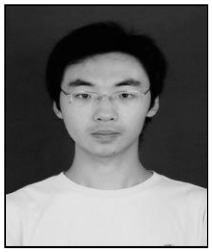
using an SDN based multicast forwarding rule and a dynamic core based tree algorithm. On the basis of this mechanism, synchronization of DSM will get a predictable response time. It is noteworthy that, we integrate advantages of SDN into inherent mechanism of distributed shared memory, but not just use SDN environment to support multicast. This is major difference between this system and other SDN multicast. SDN is a new paradigm that is not widely used in distributed system community. We believe that, there is a great potential for further research in this area. We implemented a simple prototype for SDN multicast, furthermore we plan to improve and analyze it more deeply in our future work.

References

[1] Shubin Zhang, Jizhong Han, Zhiyong Liu, Kai Wang, Shengzhong Feng, "Accelerating MapReduce with Distributed Memory Cache", Proceedings of the 15th International Conference on Parallel and Distributed Systems, Washington DC, USA, **(2009)**, December 8-11.
[2] Michiaki Iwazume, Takahiro Iwase, Kouji Tanaka, Hideaki Fujii, "Big Data in Memory: Benchmarking in Memory Database Using the Distributed Key-Value Store for Constructing a Large Scale Information Infrastructure", Proceedings of the 38th International Computer Software and Applications Conference Workshops, Vasteras, Sweden, **(2014),** July 21-25.
[3] Open Networking Foundation (ONF) , https://www.opennetworking.org/sdn-resources/sdn-definition.
[4] Kawai, E., "Can SDN help HPC?", Proceedings of the 12th International Symposium on Applications and the Internet, Izmir, Turkey, **(2012),** July 16-20.
[5] Speight E, Bennett J, "Using multicast and multithreading to reduce communication in software DSM systems", Proceedings of the 4th International Symposium on High-Performance Computer Architecture, Las Vegas, USA, **(1998),** 31 January-4 February.
[6] Speight. E, Bennett. J.K, "Brazos A Third Generation DSM System", Proceedings of USENIX Windows NT Workshop, Seattle, USA, **(1997)**, August 11-13.
[7] Takahiro Chiba, Myungryun Yoo, Takanori Yokoyama, "A Distributed Real-Time Operating System with Distributed Shared Memory for Embedded Control Systems", Proceedings of the 11th International Conference on Dependable, Autonomic and Secure Computing, Chengdu, China, **(2013)**, November 15-17.
[8] John Ousterhout, Parag Agrawal, David Erickson, Christos Kozyrakis, Jacob Leverich, David Mazières, Subhasish Mitra, Aravind Narayanan, Guru Parulkar, Mendel Rosenblum, Stephen M. Rumble, Eric Stratmann, Ryan Stutsman, "The Case for RAMClouds: Scalable High-Performance Storage Entirely in DRAM", SIGOPS Operating Systems Review, vol. 43, no. 4, **(2010)**, pp. 92-105.
[9] Po-Jen Chuang, Ting-Yi Chu, "MRBL: An Efficient Multicast Routing Protocol with Backup Labeling in MANETs", Journal of Future Generation Communication and Networking., vol. 7, no. 1, **(2014)**, pp.125-136.
[10] Harold Owens II, Arjan Durresi, "Video over Software-Defined Networking (VSDN)", Proceedings of the 16th International Conference on Network-Based Information Systems, Gwangju, Korea, **(2013)**, September 4-6.
[11] Hilmi E. Egilmez, Seyhan Civanlar, A. Murat Tekalp, "An Optimization Framework for QoS-Enabled Adaptive Video Streaming Over OpenFlow Networks", IEEE Transactions on Multimedia., vol. 15, iss. 3, **(2013)**, pp. 710-715.
[12] Siyuan Tang, Bei Hua, Dongyang Wang, "Realizing video streaming multicast over SDN networks", Proceedings of the 9th International Conference on Communications and Networking in China, Maoming, China, **(2014)**, August 14–16.
[13] Shengquan Liao, Xiaoyan Hong, Chunming Wu, Bin Wang, Ming Jiang, "Prototype for customized multicast services in software defined networks", Proceedings of the 22nd International Conference on Software, Telecommunications and Computer Networks, Split, Croatia, **(2014)**, September 17-19.
[14] Meng-Wei Lee, Yu-Sian Li, Xin Huang, Yi-Ren Chen, Ting-Fang Hou, Cheng-Hsin Hsu, "Robust Multipath Multicast Routing Algorithms for Videos in Software-Defined Networks", Proceedings of the 22nd International Symposium of Quality of Service, Hong Kong, China, **(2014)**, May 26-27.
[15] Gao Qiang, Tong Weiqin, Samina Kausar, "A High-performance DSM Leveraging Software Defined Network", Proceedings of the 44th International Conference on Parallel Processing Workshops, BeiJing, China, **(2015)**, September 1-3.
[16] Gao Qiang, Tong Weiqin, Samina Kausar, "A Design and Implementation of SDN Multicast for Distributed Shared Memory", Proceedings of the 9th International Conference on Future Generation Communication and Networking, Jeju Island, Korea, **(2015)**, November 25-28.
[17] Dashdavaa. K, Date. S., Yamanaka. H, Kawai. E, Watashiba. Y, Ichikawa. K, "Architecture of a High-Speed MPI Bcast Leveraging Software-Defined Network", Proceedings of Euro-Par 2013 Workshops, Aachen, Germany, **(2013)**, August 26-30.
[18] Beau Williamson, "Developing IP Multicast Networks", Cisco Press, Indianapolis, **(1999)**.
[19] Mininet Walkthrough, http://mininet.org/.

[20] Pantou, http://archive.openflow.org/wk/index.php/Pantou_:_OpenFlow_1.0_for_OpenWRT.
[21] OpenWrt, https://openwrt.org/.

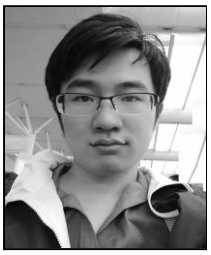
Authors



Qiang Gao, He is currently a Ph.D. student of Shanghai University. His primary research interests cover software defined network, embedded system and distributed system.

Weiqin Tong, He received his Ph.D. degree from Shanghai Jiao Tong University. He is currently a Professor of Shanghai University. His primary research interests cover high performance computing, embedded system and distributed system.

Samina Kausar, She is a Ph.D. scholar in computer science at Shanghai University China. Her research interests include distributed database systems, data mining, cloud computing and algorithms.



Shengan Zheng, He is currently a Ph.D. student of Shanghai Jiao Tong University. His primary research interests cover in-memory computing and file system.

