Distributed Active Element in 10 Gbps Network

Petr Holub and Eva Hladká
CESNET z. s. p. o., Zikova 4, 160 00 Prague, Czech Republic
Masaryk University Brno, Botanická 68a, 602 00 Brno, Czech Republic
Emails: hopet@ics.muni.cz, eva@fi.muni.cz

***Abstract*—In this paper, we propose a distributed Active Element for tightly coupled cluster environment, suitable for distribution of large bandwidth data that exceed capacity of every single node in the cluster. This approach utilizes the fact that real-time multimedia transmission systems relying on non-guaranteed protocols like UDP need to handle limited packet reordering on their own. We describe the Fast Circulating Token protocol, which enables imposing even stricter bound on the outbound packet reordering. The whole system is examined on 10GE testbed and shows very good performance. The FCT provides expected improvement, making the packet reordering comparable to long haul networks.**

***Index Terms*—multi-point user-empowered data distribution, distributed Active Elements, virtual multicast, multimedia data distribution**

I. Introduction

High-speed networking proliferation has catalyzed development and deployment of new real-time communication systems based on distribution of high-bandwidth multimedia information, like 4K and High-Definition (HD) video systems [1], [2], [3]. When designed for multi-point collaborative environments, multi-point data distribution needs to be ensured. Native network multicast has turned to perform less then acceptably in many cases and thus systems like Flexcast [1] or reflectors [2] have been developed. In this paper, we focus on a parallel reflector-based system called distributed Active Element (AE) for synchronous multimedia data distribution and processing [4], [5], which allows distribution of high bandwidth streams using parallel processing on tightly coupled commodity computer clusters. Parallel processing is incorporated in specialized hardware of modern high-performance switches and routers. However, its implementation on general purpose hardware is limited because of requirement for zero output packet reordering, which has severe consequences, e.g., on slowing down performance of the most widely used transport protocol—TCP [6]. The real-time multimedia communication relying on non-guaranteed transmission protocols, like RTP over UDP, however needs to handle the packet reordering on their own anyway, as the reordering is often present in long-distance transmissions for high-bandwidth data rates [6], [7]. We examine performance of the distributed AE in 10 Gigabit Ethernet environment for distribution of multi-Gigabit data streams, as used by advanced multimedia systems relying on uncompressed HD and post-HD video.

Related work. Since the native multicast support is not always available, reliable, or performing well enough, multicast virtualization technologies have been introduced, like the H.323 MCUs or reflectors in the Virtual Room Videoconferencing System (VRVS)[1]. Its successor EVO [8] is based on self-organization of system of reflectors. Similar approach has been pursued earlier by our group [9]. Other simpler UDP packet reflectors include rcbridge [10], reflector[2], and Alkit Reflex[3].

Another area related to this paper is utilization of computer clusters as either distributed routers or distributed servers [11], [12], [13]. Project Suez [14] is a distributed router based on commodity PC cluster with Myrinet interconnection with each node of the cluster having one internal interface to the Myrinet switch and optionally one or more external interfaces. Suez uses a routing-table search algorithm that exploits CPU cache for fast lookup by treating IP addresses directly as virtual addresses. Another project which distributes processing load on active network elements is Active Network Node [15] that relies on specialized hardware. Software DSM project [16] attempts to build efficient distributed memory for closely coupled clusters for using them as active routers. There is yet another similar project called Cluster-based Active Network Router [17]. However none of the above mentioned projects addresses finer than per-address network load distribution—thus there is no need for solving packet reordering issues, but on the other hand it doesn't solve the problem of data distribution for streams where the bandwidth exceeds capacity of each single parallel unit in the cluster.

Our distributed AE is designed not only for data distribution, but allows for processing as well. Relevant parallel programming paradigm for stream processing on distributed clusters has been proposed in MIT StreamIt project [18].

Paper organization. Section II gives a brief overview of distributed AE architecture, introducing the packet reordering limiting using Fast Circulating Token protocol. Section III describes experimental results of distributed AE used with multi-Gigabit data flows. Concluding remarks and future work ideas are given in Section IV.

II. Distributed Active Element

In this section, we give an overview of the distributed AE and its basic properties with respect to data distribution, required for understanding experimental section of this paper. More details on the distributed AE can be found in [4], [5]. The architecture of the distributed AE is based on architecture

[1] http://www.vrvs.org/
[2] http://www.cs.ucl.ac.uk/staff/s.bhatti/teaching/z02/reflector.html
[3] http://w2.alkit.se/reflex/

of AE [9] and it is partly determined by requirement of implementability on existing tightly coupled clusters with low latency interconnection. The computing nodes form a computer cluster with each node having two connections: (1) *low-latency control connection* used for internal communication and synchronization inside the distributed AE, and (2) *data connection* used for receiving and sending the data. Example of such a system is shown in the evaluation testbed description in Figure 2.

The incoming data needs to be first distributed across the multiple parallel units of the distributed AE, processed in these units, and finally aggregated and sent over the network to the listening clients. Thus the architecture comprises three major parts:

- *Distribution unit* takes care of ingress data flow distribution over multiple parallel distributed AE units. When the distribution unit is part of the same L2 domain as parallel AE units (e.g., using VRRP or CARP protocols), it may operate on L2 addresses only, otherwise L3 (usually IP) addressing is needed.
- *Parallel AE unit* is a complete instance of AE with modified sender module to allow for possible synchronization. It has the kernel with administrative submodules, session management, processor schedulers, and AAA submodules. Data is received by network listener modules, stored into shared memory (shared across a single instance of the reflector only, not across multiple AE unit instances), processed by zero or more processors, distribution lists filled up with either one of processors or with session management and finally sent with the sender module. The network management module handles communication with distribution unit and also communication with other distributed AE units.
- *Aggregation unit* aggregates the resulting traffic to output network line(s). Because the AE element is most commonly used for data multiplication, we assume that output data flow from the distributed AE is larger than input data flow. Thus we need a unit that is even more powerful than the input load distribution unit. In most cases, cheap custom made software implementation is not available and we have to use available hardware solution like aggregating switch. However, in that case we must not assume any further behavior of the aggregating unit except for the following two things: first, it is over-provisioned enough not to loose any data, and second, it has limited buffer space available.

There are also protocols designed for set up and maintenance the distributed AE [5], so that new parallel units may join in and existing may leave.

- The *ideal network* is a network in which no data is lost, corrupted, nor reordered. It also provides instant delivery, i. e., it introduces zero latency.
- The *ideal multimedia traffic* has bandwidth b and independent packets of exactly same size s_p, which is also used to express all the queue sizes in the system. In order to isolate reordering introduced by the distributed AE, we assume that the ideal multimedia traffic has no reordering prior to entering distribution unit.
- The *ideal aggregating unit* has n identical input interfaces and a single output interface with capacity equal or bigger than the n inputs together. It reads packets from the size-limited input interface queues and sends them on an output interface in such a way, that packets are never lost. The speed is b_j^{SW} (bits per second) for j-th input interface and each input queue has equal size of s_i^{SW} for each input interface. In order not to lose any input data, the ideal aggregating unit needs to fulfill the following requirement in the steady state: $\sum_j b_j^{SW} \leq b_o^{SW}$.
- The *ideal AE* has processing capacity equal or higher than stream bandwidth and it has an input queue size of s_i^{AE} and all the parallel units have the same parameters and performance. The ideal AE introduces no losses, no data corruption, nor data reordering in the data stream.

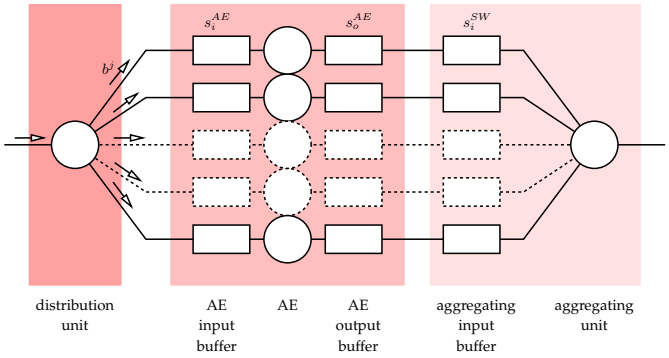

Figure 1. Model of the ideal distributed AE with ideal aggregation unit.

A. Ingress Distribution

The ingress data distribution takes care of distributing incoming data across different paths inside the distributed AE. For the ideal distributed AE the ideal distribution unit distributes packets in round-robin fashion. In each round, it distributes m packets, one to each of the parallel units. The distribution unit marks round number into each packet. This protocol is modified in case that parallel units are of unequal performance.

B. Egress Synchronization

1) No explicit synchronization: The simplest model for egress synchronization is to use no synchronization at all. However, with this model and limited buffers on the input interfaces of the AEs, there is still some implicit synchronization achieved.

It can be shown the maximum reordering induced by an ideal distributed AE (shown in Figure 1) with no explicit egress synchronization and ideal aggregating unit is

$$n(s_i^{AE} + s_o^{AE} + s_i^{SW} + 1),$$

where n is the number of parallel AE units when all queues operate in FIFO tail-drop mode. Detailed proof can be found in [5].

2) Fast Circulating Token: In order to decrease packet reordering introduced by the distributed AE, we have introduced a distributed algorithm for achieving less packet reordering compared to no explicit synchronization. The nodes are ordered in a ring with one node elected as a master node and they circulate a token which serves as a barrier so that no node can run too much ahead with sending data. After reception of the token containing the current "active" round number, each non-master node passes on the token immediately and may send only the data from the round marked in the token until it receives to token again. When the master node receives the token from the last node in the ring, it finishes sending the current round, increments the round number in the token a passes on the token. The mechanism is called *Fast Circulating Token* (FCT) since the token is not held for the entire time period of data sending as usual in the token ring networks.

Because of real world implementation of data packet sending in common operating systems, we assume that sending procedure for a single packet is non-preemptive. Further we assume that token reception event processing has precedence over any other event processing in the distributed AE. However, as the data sending is non-preemptive, if the token arrives in the middle of data packet sending, it will be handled just after that packet sending is finished.

After more detailed analysis [5], it can be shown the maximum reordering induced by an ideal distributed AE with FCT egress synchronization and ideal aggregating unit is

$$n(s_i^{SW} + 3),$$

where n is the number of parallel AE units when all queues operate in FIFO tail-drop mode.

When operating in a non-ideal environment, there are several complications that need to be taken into account:

- packet reordering, either before data reaches the distributed AE, or on a single parallel path inside distributed AE—settings of the FCT determine whether excessive packet reordering will be converted to packet loss or not,
- due to unequal performance of parallel paths, load balancing may be deployed—again the reordering of two consecutive packet is limited by size of two consecutive rounds, but each round may have more than n packets depending on load balancing scheme used,
- packet loss due to overloading of distributed AE or some of its parts.

3) Exact Order Sending: It is possible to design sending protocol that results into exact ordering, but it requires defined behavior of aggregation unit and thus it is not suitable for implementation on commodity hardware like aggregating switches. The details can be found in [4], [5].

III. Prototype Performance Evaluation in 10GE Environment

Prototype implementation of the distributed AE is implemented in ANSI C language for portability and performance reasons. The implementation comprises two parts: a load distribution library and the distributed AE itself.

Because of lack of flexible enough load distribution hardware unit, we have implemented it as a library, which allows simple replacement of standard UDP related sending functions in existing applications and allows developers to have defined type of load distribution—either pure round robin or load balancing.

Each parallel AE uses threaded modular implementation based on architecture described in Section II. Internal buffering capacity of each AE node has been set to 500 packets. Explicit synchronization using FCT protocol has been implemented using MPICH implementation[4] of MPI built with low-latency Myrinet GM 2.0 API[5] (so called MPICH–GM). Prototype implementation has been tested on Linux.

For cost-effective prototype implementation, the aggregation unit was a implemented as commodity switch with sufficient capacity of internal switching matrix.

A. Experimental Setup

The behavior of the distributed AE has been evaluated on a 10GE testbed shown in Figure 2. The sender and receiver machines were identical PCs with dual AMD Opteron 250 processor at 2.4 GHz, 2 GB RAM, and 10GE Chelsio T110 NIC card in a PCI-X 133 MHz slot. Both computers were running SuSE Linux 9.1 with 2.6.6 vanilla kernel with Chelsio drivers and patches. The computers were connected to 10GE ports of the Cisco 6506 switch.

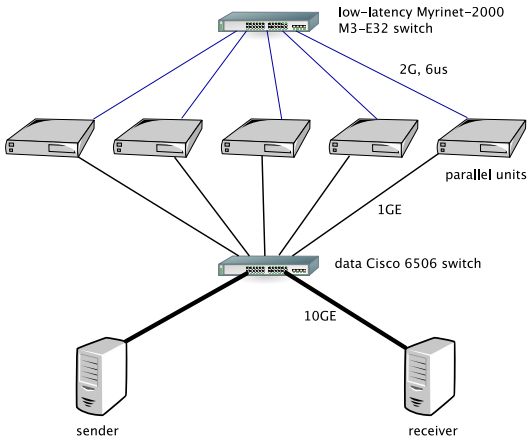

Figure 2. Experimental testbed setup.

The parallel AE units were run on a IA32 PC cluster with Myrinet-2000 low-latency interconnection. The nodes were equipped with dual Intel Xeon at 2.4 GHz, 2 GB RAM, and Broadcom NetXtreme BCM5703 card plugged into Gigabit

[4]http://www-unix.mcs.anl.gov/mpi/mpich/
[5]http://www.myri.com/scs/GM-2/doc/html/

# parallel units	*max. bw [Mbps]*
1	800
2	1600
3	2400
4	3200
5	4000
6	5000
7	5000
8	5000

Table I
MAXIMUM FORWARDING BANDWIDTH FOR VARYING NUMBER OF PARALLEL UNITS FOR THE DISTRIBUTED AE.

Ethernet ports of Cisco 6506 switch. Each node had also Myrinet M3F-PCI64C-2 NIC plugged into Myrinet M3-E32 with M3-SW16-8F interface.

The distribution unit was implemented in software based on direct IP addressing of the parallel nodes, the stream splitting was carried out by the 6506 switch, which was also acting as the aggregating unit.

B. Performance Evaluation

In order to evaluate raw performance of the AE, we have measured maximum bandwidth of a stream that the AE is able to forward without packet loss greater than 0.1%. This benchmark more demanding and thus also representative because of memory to memory copying limitations, compared to the replication, which easily saturates the bandwidth of the outgoing network interface [19].

a) Stand-alone reflector: Standalone reflector running on one node of the testbed cluster was measured for reference purposes and it was able to forward data up to 800 Mbps.

b) Distributed AE: Maximum bandwidth for forwarding with less than 0.1% packet loss was measured for varying number of parallel units and the results are summarized in Table I. Note the saturation at 6 parallel units, which is caused by the maximum throughput on the Chelsio T110 cards on sender and receiver.

For low loss area, the results are equivalent both for the unsynchronized and FCT-synchronized version of the distributed AE. It also turns out that the performance scales linearly with respect to the number of parallel AE units. When examining the higher loss areas (which are not usable for real data distribution anyway), the FCT-synchronized version performs slightly worse than the unsynchronized. This can be observed from upper part of the graphs in Figure 3, for 2 and 4 parallel units respectively; the lines are overlapping for 6 and 8 units as the higher loss region is not reached because of sender/receiver saturation as discussed above.

C. Packet Loss and Reordering Evaluation

The reordering is expressed as the difference between sequence numbers of two consecutive packets. Thus, if all the sequentially numbered packets arrive in the same order they were sent, all the differences are +1. Higher number than +1 means, that some packets were skipped forth (either because

BW	FCT-sync			unsync		
[Mbps]	$\min\{j\}$	H^-	N^-	$\min\{j\}$	H^-	N^-
200	0	0	0	-4	-12	3
400	0	0	0	-6	-7	2
600	-1	-6	6	-3	-5	2
800	-5	-26	16	-4	-13	5
1000	-6	-124	93	-10	-21	5
1200	-7	-82	67	-33	-113	15
1400	-5	-220	179	-39	-455	65
1600	-6	-522	466	-7	-301	281
1800	-5	-1162	1104	-50	-911	627
2000	-5	-1317	1252	-20	-1214	1119
2200	-7	-1545	1443	-10	-1706	1599
2400	-7	-2634	2520	-14	-2591	2435
2600	-6	-4946	4736	-14	-5423	5177
2800	-15	-6539	5886	-15	-7351	7005
3000	-6	-7963	7424	-31	-10987	9654
3200	-7	-8712	7117	-89	-9420	7592
3400	-7	-9431	5060	-100	-9104	4827
3600	-7	-57523	27730	-38	-50178	26252
3800	-7	-256152	122298	-111	-253093	125121
4000	-7	-482062	229886	-23	-480988	236434
4200	-7	-989134	464849	-33	-952629	476268
4400	-7	-1484894	685827	-46	-1535218	755956
4600	-7	-1081210	497005	-25	-938601	473483
4800	-7	-406902	183817	-43	-181140	97237
5000	-7	-38077	19546	-27	-13435	10480

Table II
PACKET REORDERING FOR 8 PARALLEL UNITS.

of packet reordering or because of packet loss) while negative number means stepping back in packet numbering (due to packet reordering only). Value of 0 occurs when duplicate packets arrive immediately following each other. $\min\{j\}$ is the maximum negative difference in sequence numbers of successively received packets. The $\min\{j\}$ is very important from the application developer perspective, as it gives the amount of packet buffer needed to reconstruct the proper order of packets (provided no packet loss occurs), and also from the users perspective, as the amount of buffering it related to latency increase the users are experiencing.

For any interval of arrivals of two or more packets, the following equation holds

$$\underbrace{\sum_{j=\min\{j\}}^{-1} jh_j}_{H^-} + \underbrace{h_1}_{H^1} + \underbrace{\sum_{j=2}^{\max\{j\}} jh_j}_{H^+} = \Delta, \qquad (1)$$

where Δ is difference between sequence number of last and first packet in the observed interval. Also, for the any interval of arrivals of more than one packet, the following equation holds:

$$\Pi + \underbrace{\sum_{j=\min\{j\}}^{-1} h_j}_{N^-} + \underbrace{h_1}_{N^1} + \underbrace{\sum_{j=2}^{\max\{j\}} h_j}_{N^+} - \delta = \Delta. \qquad (2)$$

where Π is number of lost packets and δ is a number of duplicated packets that are not included in h_0. Proofs for both can be found in [5]. By combining both equations (1) and (2), we can derive packet loss as $\Pi = H^- + H^+ - N^+ - N^- + \delta$. Because positive part of the graph described by H^+ or N^+ includes also packet loss, the negative part of the graph

described by H^- or N^- can be seen as measure of packet reordering.

The difference between the H-sums and the N-sums is that the H-sums are "weighted sums". Thus the more packets are farther from 1 in either direction, the higher the absolute value of H-sums are, while the N-sums remain the same. All the terms in the N^-, N^+, and H^+ are positive and all the terms in the H^- are negative. If $H^- \approx N^-$, the vast majority of out-of-order packets in the negative part is reordered by $j = -1$.

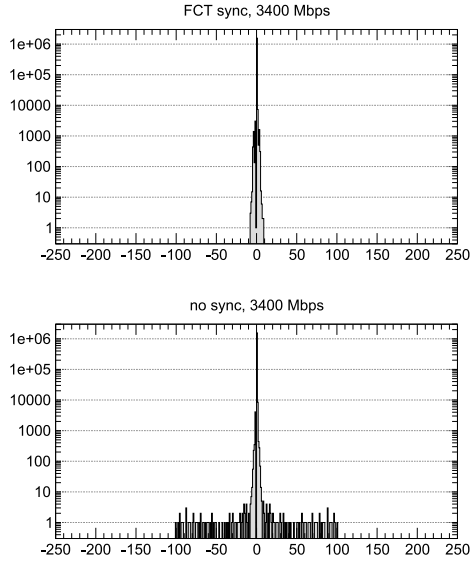

Figure 4. Sample packet reordering distribution with FCT and without synchronization, for 8 parallel units and 3.4 Gbps per data flow.

c) Stand-alone reflector: The bigger the loss is above the 800 Mbps performance limit, the larger the sum H^+ is. No reordering nor duplicates are introduced, thus $H^- = N^- = h_0 = 0$.

d) Distributed AE: The results presented here are a subset of complete set of measurement carried out in order to evaluate the behavior of distributed AE throughly. Figure 4 shows dependence of $\min\{j\}$ on bandwidth of the forwarded stream (lower part of each graph) together with packet loss (upper part) for 2, 4, 6, and 8 parallel units. The behavior for 3, 5, and 7 is comparable. It turns out that before the saturation of the distributed AE (c.f. Table I), the FCT significantly improves maximum packet reordering $\min\{j\}$.

More detailed results for 8 parallel units is shown in Table II, revealing that H^- and N^- are similar for FCT and unsynchronized versions. Sample reordering distribution for the 3400 Mbps stream and 8 parallel units is given in Figure 4. The reordering results are appropriate to be viewed in the context of long haul network paths. Given examples [7] of rather problematic Washington D. C. to Los Angeles link with $\min\{j\}$ over -60 and high-quality link from Los Angeles to Pittsburgh with $\min\{j\}$ of -1, the FCT gives much better results than the former link and gives comparable results to the latter one. Thus a real-time multimedia transmission application, which has been developed to work over long distance networks, needs to adapt to even significantly worse packet reordering than the one outgoing from distributed AE, to perform reliably in real world conditions.

Token round-trip time in FCT protocol has been also monitored and it ranges between 14 μs for 2 parallel AE paths and raises up to approximately 60 μs for 8 parallel paths, which is in accordance with one-way message passing latency of Myrinet configuration used for the testbed as described above.

IV. Conclusions

In this paper, we have presented the concept of distributed Active Element, which relaxes the requirement for strict packet ordering, assuming that the real-time multimedia transmission applications relying on non-guaranteed protocols like UDP need to adapt to some degree of packet reordering on their own. This allowed us to design and prototype a scalable system for data distribution and potentially also processing of real-time multimedia data based on tightly coupled clusters with low-latency internal interconnection. We have proposed the Fast Circulating Token protocol in order to impose harder upper bound on the maximum packet reordering. The prototype system has been examined using 10 Gigabit Ethernet testbed and the results suggest its usability for high-end applications.

In the future, we would like to focus on three basic areas. First, we would like to adapt some of the parallel stream processing paradigms like StreamIt [18] to program data processing for the distributed AE, thus turning the system into a more general active router. Second, having the data processing, we would like to extend the distributed AE with quality of service support on several levels (network bandwidth, CPU capacity, memory capacity and bandwidth, etc.). Third, we intend to examine suitability of custom hardware solutions based on FPGA to build a aggregation unit allowing exact order packet sending.

Acknowledgments

This project has been supported by a research intent "Optical Network of National Research and Its New Applications" (MŠM 6383917201) and "Parallel and Distributed Systems" (MŠM 0021622419). The authors would like to acknowledge help of Jiří Denemark and Tomáš Rebok for their assistance during measurements implementation.

References

[1] T. Shimizu, D. Shirai, H. Takahashi, T. Murooka, K. Obana, Y. Tonomura, T. Inoue, T. Yamaguchi, T. Fujii, N. Ohta, S. Ono, T. Aoyama, L. Herr, N. van Osdol, X. Wang, M. D. Brown, T. A. DeFanti, R. Feld, J. Balser, S. Morris, T. Henthorn, G. Dawe, P. Otto, and L. Smarr, "International real-time streaming of 4K digital cinema," *Future Generation Computer Systems*, vol. 22, no. 8, pp. 929–939, Oct. 2006.

[2] P. Holub, L. Matyska, M. Liška, L. Hejtmánek, J. Denemark, T. Rebok, A. Hutanu, R. Paruchuri, J. Radil, and E. Hladká, "High-definition multimedia for multiparty low-latency interactive communication," *Future Generation Computer Systems*, vol. 22, no. 8, pp. 856–861, 2006.

[3] J. Jo, W. Hong, S. Lee, D. Kim, J. Kim, and O. Byeon, "Interactive 3D HD video transport for e-science collaboration over UCLP-enabled GLORIAD lightpath," *Future Generation Computer Systems*, vol. 22, no. 8, pp. 884–891, 2006.

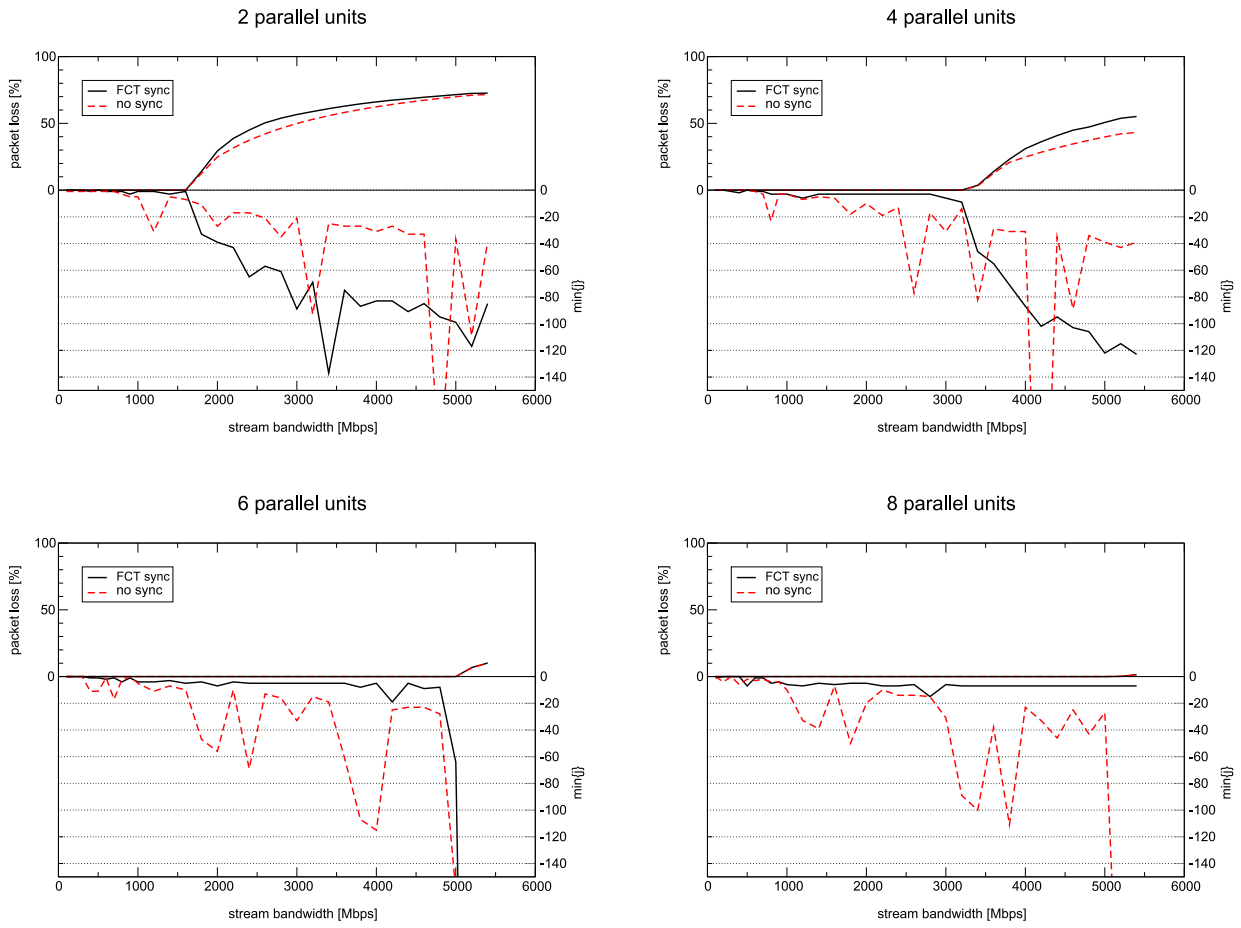

Figure 3. Packet loss and reordering comparison for FCT synchronized and unsynchronized version for distributed AE and various number of parallel units.

[4] P. Holub and E. Hladká, "Distributed active element for high-performance data distribution," in *NPC 2006: Network and Parallel Computing*, Tokio, Japan, Oct. 2006, pp. 27–36.

[5] P. Holub, "Network and grid support for mulitmedia distribution and processing," Ph.D. dissertation, Faculty of Informatics, Masaryk University Brno, Czech Republic, 2005.

[6] J. C. R. Bennett, C. Partridge, and N. Shectman, "Packet reordering is not pathological network behavior," *IEEE/ACM Transactions on Networking*, vol. 7, no. 6, pp. 789–798, Dec. 1999.

[7] L. Gharai, C. Perkins, and T. Lehman, "Packet reordering, high speed networks and transport protocol performance," in *Proceedings of the 13th International Conference on Computer Communications and Networks (ICCCN'04)*, Chicago, IL, USA, Oct. 2004, http://ultragrid.east.isi.edu/publications/2004icccn.pdf.

[8] P. Galvez, "From VRVS to EVO (Enabling Virtual Organizations)," in *TERENA Networking Conference 2006*, Catania, Italy, May 2006.

[9] P. Holub, E. Hladká, and L. Matyska, "Scalability and robustness of virtual multicast for synchronous multimedia distribution," in *Networking - ICN 2005: 4th International Conference on Networking, Reunion Island, France, April 17-21, 2005, Proceedings, Part II*, ser. Lecture Notes in Computer Science, vol. 3421/2005. La Réunion, France: Springer-Verlag Heidelberg, Apr. 2005, pp. 876–883. [Online]. Available: http://www.springerlink.com/index/GETV62MG4GA0CUPL

[10] M. Buchhorn, "Designing a multi-channel-video campus delivery and archive service," in *The 7th Annual SURA/ViDe Conference*, Atlanta, GA, USA, Mar. 2005.

[11] V. S. Pai, M. Aron, G. Banga, M. Svendsen, P. Druschel, W. Zwaenepoel, and E. Nahum, "Locality-aware request distribution in cluster-based network servers," in *Proc. Eighth ACM Conf. Architectural Support for Programming Languages and Operating Systems*, Oct. 1998, pp. 205–216.

[12] R. Bianchini and E. V. Carrera, "Analytical and experimental evaluation of cluster-based WWW servers," *World Wide Web J.*, vol. 3, no. 4, pp. 215–229, Dec. 2000.

[13] E. V. Carrera and R. Bianchini, "Press: A clustered server based on user-level communication," *IEEE Transactions on Parallel and Distributed Systems*, vol. 16, no. 5, pp. 385–395, May 2005.

[14] T. Chiueh and P. Pradhan, "Suez: A cluster-based scalable real-time packet router," in *The 20th International Conference on Distributed Computing Systems (ICDCS 2000)*, Taipei, Taiwan, Apr. 2000.

[15] D. Decasper, G. Parulkar, and B. Plattner, "A scalable, high performance active network node," *IEEE Network*, vol. 33, no. 1, pp. 8–19, Jan. 1999.

[16] P. Graham, "A DSM cluster architecture supporting aggressive computation in active networks," in *Intl. Workshop on Distributed Shared Memory*, 2001. [Online]. Available: http://www.cs.umanitoba.ca/~pgraham/papers/DSM_body.pdf

[17] Y. B. Jang and J. W. Cho, "A cluster-based router architecture for massive and various computations in active networks," in *ICOIN*, KAIST, Korea, Feb. 2003. [Online]. Available: http://camars.kaist.ac.kr/~ybjang/research/publication/icoin2003.pdf

[18] W. Thies, M. Karczmarek, and S. Amarasinghe, "Streamit: A language for streaming applications," in *Proceedings of the 11th International Conference on Compiler Construction*, ser. Lecture Notes in Computer Science, vol. 2304/2002. Grenoble, France: Springer-Verlag Heidelberg, 2002, pp. 179–196.

[19] E. Hladká, "User empovered collaborative environment: Active network support," Ph.D. dissertation, Faculty of Informatics, Masaryk University Brno, Czech Republic, 2004.