

Distributed Active Element for High-Performance Data Distribution

Petr Holub^{1*}

Eva Hladká^{2†}

¹*Institute of Computer Science
Masaryk University,
Botanická 68a, 602 00 Brno, Czech Republic*

²*Faculty of Informatics,
Masaryk University,
Botanická 68a, 602 00 Brno, Czech Republic*

1 Introduction

Requirement on multi-point data distribution in IP networks assumes some distribution service, be it implemented as a part of network native services (IP multicast) or user-empowered solution (data reflectors). In our previous work, we have introduced user-empowered distribution networks based on Active Elements (AE) [1], which scale well in terms of number of clients connected. It is however not sufficient in terms of scalability with respect to bandwidth of each single stream distributed, i.e. it is not suitable for distributing streams whose bandwidth exceeds capacity of each single AE [2].

In order to improve the scalability with respect to the bandwidth of a stream, we propose a concept of distributed AE, suitable for implementing on computer clusters with low-latency internal interconnection. Its architecture is based on parallelizing whole AE architecture including listener and sender modules, which however brings problems with packet reordering in sending part. While packet reordering is largely unwanted for general network element (e.g. router) behavior, as it severely tampers performance especially of TCP-based applications, it is more acceptable for multimedia application that rely on UDP protocol and thus need to handle potential packet reordering anyway. Simple solution is a design with all distributed modules except for sender module—while this would help for computationally intensive operations on the streams, it is of little use for high-bandwidth streams. In this paper we show, that even when having multiple sending modules with no explicit synchronization, the reordering introduced by the distributed AE has an upper bound for real-time synchronous applications under certain assumptions. It can be further reduced by implementing proposed Fast Circulating Token protocol.

Related work. Distribution of multimedia data over IP network leads to a multicast schema. However, as the native multicast solution is not always reliable or even available, other distribution schemes were developed following approach of multicast virtualization. They are usually based on a central distribution unit—a reflector—

like the H.323 MCUs or reflectors provided in the Virtual Room Videoconferencing System (VRVS)¹. The successor of VRVS called Enabling Virtual Organizations (EVO)[3] is based on self-organization of system of reflectors, again not empowering the end-user with tools to change the distribution topology. Other simpler UDP packet reflectors include rcbridge [4], reflector², and Alkit Reflex³.

Another area related to this paper is utilization of computer clusters as either distributed routers or distributed servers. Project Suez [5, 6] 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 [7] that relies on specialized hardware. Software DSM project [8] attempts to build efficient distributed memory for the closely coupled clusters for using them as active routers. There is yet another similar project called Cluster-based Active Network Router [9]. However none of the above mentioned projects addresses finer than per-address network load distribution and thus there is no need for solving packet reordering issues.

There is a number of distributed servers based on employing computer clusters. Most distributed servers are prototyped as web servers [10, 11, 12] for simplicity reasons and because rather standard and straightforward performance evaluation is available. For example, Carrera and Bianchini recently demonstrated cluster based web server called PRESS [13] concentrating on demonstrating advantages of user level communication like low processor overhead, remote memory accesses, and zero-copy transfers.

Paper organization. This paper is organized as follows. Section 2 discusses general architecture of the distributed AE and its behavior and the packet reordering

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/>

*hopet@ics.muni.cz

†eva@fi.muni.cz

from the theoretical point of view. Prototype implementation is described and evaluated in Section 3. The paper concludes with future work and concluding remarks in Section 4.

2 Distributed Active Element

We are proposing the distributed AE based on architecture of AE described in [1] and it is partly determined by requirement of implementability on existing tightly coupled clusters with low latency interconnection. The distributed AE implementation assumes the infrastructure as shown in Figure 1. 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. Low latency interconnection is necessary since current common network interfaces like Gigabit Ethernet provide large bandwidth, but latency of the transmission is still in order of hundreds of μs , which is not suitable for fast synchronization. Specialized low-latency interconnects like Myrinet provide as low latency as $6 \mu s$, which is comparable to message passing between threads on a single computer.

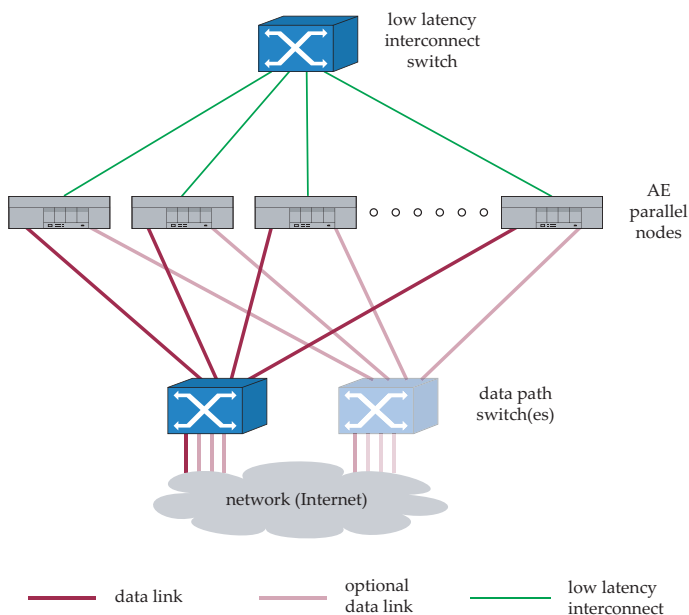


Figure 1: Model infrastructure for implementing the distributed AE.

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 send 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, it may operate on L2 addresses only (e.g. Ethernet ad-

resses in operation similar to VRRP⁴ or CARP⁵ protocols), 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 are received using network listener modules, stored into shared memory (shared across the 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. Unless there is some complex data processing involved, data passes through distributed AE unit in zero copy mode for performance reasons.

The network management module handles communication with distribution unit and also communication with other distributed AE units if AE ring is to be set up and maintained for the Fast Circulating Token protocol (Section 2.2). However, handling token itself for this protocol is performed directly by the sender module in order to minimize operation overhead.

- *Aggregation unit* aggregates the resulting traffic to output network line(s). Because the AE element is often used as data multiplication unit, 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 and 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 two things: first, it is over-provisioned enough not to lose any data and second, it has limited buffer space available.

Whole architecture supports user-empowered operation, there is still no need for running any part of it in kernel space. The only administrative requirement is that the cluster environment needs to be set up together with networking infrastructure including distribution and aggregation units.

In order to set up and maintain the distributed AE, some protocol is needed – it has been described in [14] in detail and it is out of scope of this paper. In the rest of the paper we describe operation of distributed AE in static environment, where there is constant number of AEs participating in distributed processing and the AEs work reliably.

In order to evaluate our models theoretically, we need to introduce an idealized environment:

⁴<http://www.ietf.org/html.charters/vrrp-charter.html>

⁵<http://www.countersiege.com/doc/pfsync-carp/>

- The *ideal network* is a network in which no data are 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 equal or smaller than MTU of the underlying network. The packets are sent in regular intervals. All the queue sized below are expressed in units of packet size s_p . 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 input interfaces with the same parameters and one output interface with capacity equal or bigger than the n inputs together. It reads packets from the size-limited input interfaces queues and sends them on output interface in such a way, that packets are never lost. The speed is b_j^{SW} 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 q_i^{AE} . All the parallel units of the ideal AE have the same parameters and performance and the total bandwidth of the traffic is divided into streams with the same parameters. The ideal AE introduces no losses, nor data corruption, nor data reordering in the data stream.

2.1 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 n packets, one to each of the parallel units. The distribution unit marks round number into each packet.

Such an ideal distribution might not be suitable in the cases, when parallel AE units are of unequal performance or when data stream packets are not independent and the processing needs to have all the inter-dependent packets through the same path. When parallel AE units do not have the same performance, the load balancer can send multiple packets in each round to the same parallel path. All the packets sent in one round are marked with the same round number. Sample packet distribution is shown in Figure 3.

2.2 Egress Synchronization

2.2.1 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 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 [14].

2.2.2 Fast Circulating Token.

In order to decrease packet reordering introduced by the distributed AE, we propose 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. 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 is non-preemptive, i. e. once a packet is being sent, this process can be interrupted after sending is finished. Further we assume that token reception event processing has precedence over any other event processing in the distributed AE. If there are multiple token events waiting, they are processed in FIFO way. 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.

The token carries the following information: (1) *round number* – corresponds to round number from distribution unit, which is set and incremented on master node, (2) *last round-trip time*, and (3) *holding time left after traveling from master to current node*

Depending on implementation circumstances, `timeLeft()` function may be used to allow keeping token on other nodes than master for limited amount of time. This might be needed if e.g. the master node is considerably faster than other nodes. For ideal distributed AE, `timeLeft()` returns 0.

After more detailed analysis [14], 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 needs to be taken into account:

- packet reordering, either before data reach distributed AE, or on a single parallel path inside distributed AE – possible implementation of the first condition after token reception influences 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

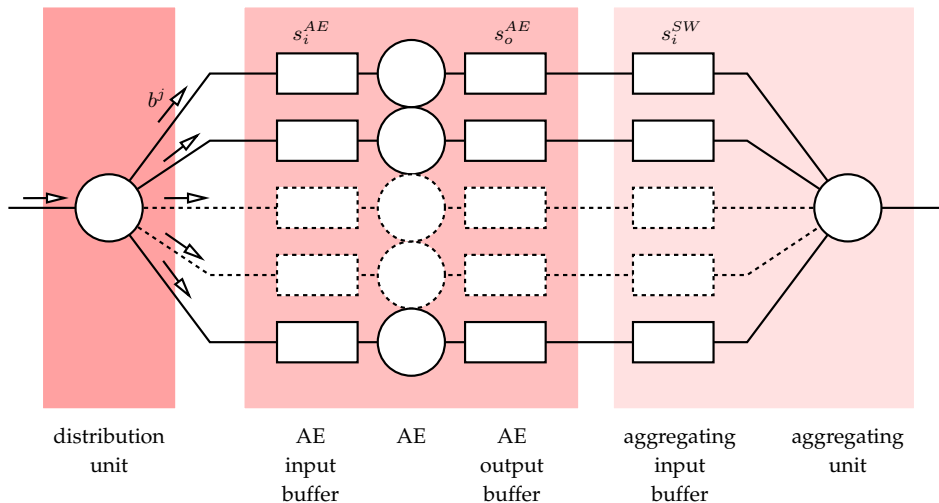


Figure 2: Model of the ideal distributed AE with ideal aggregation unit.

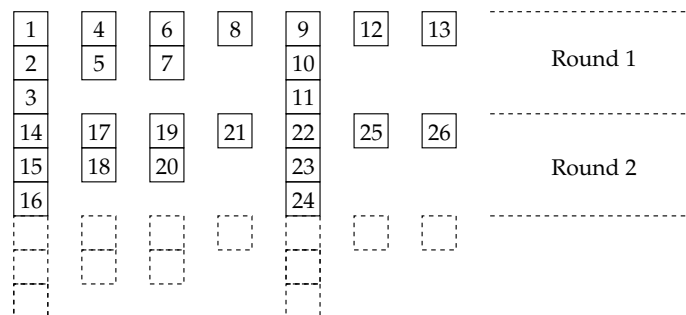


Figure 3: Sample load balancing packet distribution for distributed AE.

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.

2.2.3 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.

One possibility is that aggregation unit behaves similar to sending modules with FCT protocol, i. e. it reads packets from input either in the same round robin way distribution unit distributes packets and utilizes mark of round number in each packet to recognize when the packet is ready to be sent. When each parallel path is ideal, namely there is no packet loss or corruption, even packet round numbering might not be necessary. However, in order to protect aggregation against lost and/or corrupted packets which would make one of the queues go ahead of the rest, it is advisable to stick to round numbering of the packets. For such operation, an alternative packet distribution shown in Figure 5 is more appropriate—compared to the distribution shown in Figure 3, it has smaller rounds containing either zero or one packet. Instead of sending no packet, it actually needs

to send empty round marker to let the aggregator know that there is no need to wait for the packet to arrive. It not efficient when circulating token is present as there are more rounds and thus it pronounces overhead of the token.

Such protocol might be implementable on custom hardware, e. g. data switch, or custom network processor, or at least some programmable routing device like FPGA-enabled routing cards.

3 Prototype Performance Evaluation

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 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 element uses threaded modular implementation based on architecture described in Section 2. Internal buffering capacity of each AE node has been set to 500 packets. Explicit synchronization using

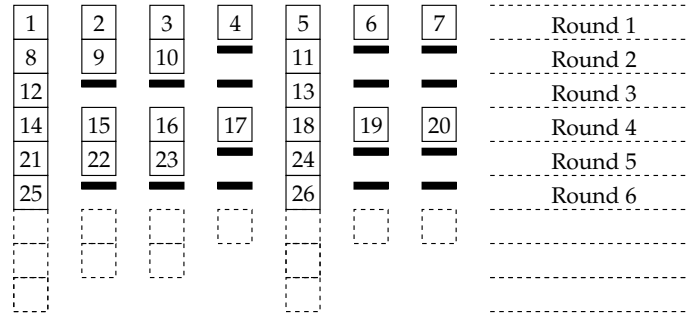


Figure 5: Alternative load balancing packet distribution. Empty round markers are shown as small black filled rectangles.

```

1 rnd := 0;
2 finish := false;
3 while ¬finish do
4   if ¬(master ∧ rnd = 0)
5     token_ready := 0;
6     do
7       if test_rcv_token(rnd_no, last_RTT, t_left)
8         token_ready := 1;
9         if ¬master ∧ t_left = 0
10          pass_token(rnd_no, last_RTT, t_left);
11        fi
12      fi
13      if test_rcv_finish()
14        finish := true;
15      fi
16      send_packet(rnd);
17      while ¬(token_ready ∨ finish) od
18      if finish
19        break;
20      fi
21    fi
22    if master
23      send_all_packets(rnd);
24      rnd := get_rnd_from_queue();
25      last_RTT := updateRTT();
26      t_left := timeLeft();
27      pass_token(rnd, last_RTT, t_left);
28    else
29      while t_left > 0 ∧ is_packet(rnd) > 1 do
30        send_packet(rnd);
31        t_left := t_left - 1;
32      od
33      pass_token(rnd_no, last_RTT, t_left);
34      discard_packets(rnd);
35      rnd := rnd_no;
36    fi
37    if master ∧ finish_requested
38      foreach s ∈ slaves do
39        send_finish(s);
40      od
41      finish := true;
42    fi
43 od

```

Figure 4: Fast Circulating Token algorithm.

FCT protocol has been implemented using MPICH implementation⁶ of MPI⁷ built with low-latency Myrinet GM 2.0 API⁸ (so called MPICH-GM). Prototype implementation has been tested and known to work on Linux.

For cost-effective prototype implementation, the aggregation unit was implemented as commodity switch satisfying condition that egress link capacity is equal or larger than ingress capacities and with sufficient capacity of internal switching matrix.

3.1 Experimental Setup

In order to evaluate performance and behavior of the distribute AE experimentally, we have set up a testbed shown in Figure 6 comprising eight machines and two switches:

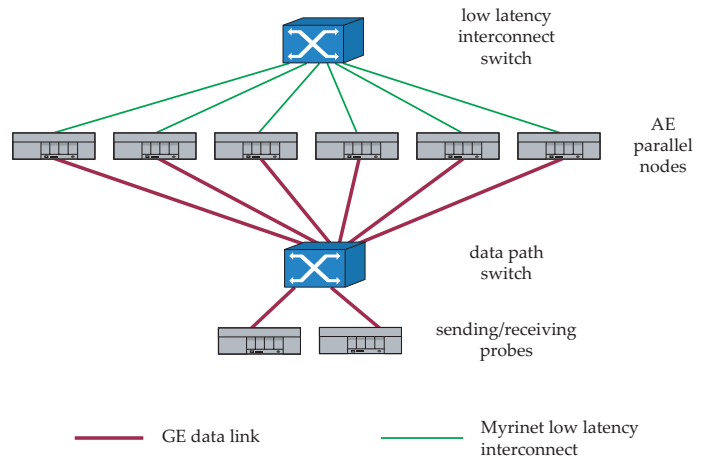


Figure 6: Distributed AE testbed setup.

- A data switch HP ProCurve 6108 with 8× Gigabit Ethernet full wire-speed ports. Manufacturer-specified switching capacity is 16 Gbps and switching performance of 11.9 million packets per second is given for 64 B packets.
- A low-latency Myrinet M3-E32 switch with M3-SW16-8F interface cards that created the control

⁶<http://www-unix.mcs.anl.gov/mpi/mpich/>

⁷<http://www.mpi-forum.org/>

⁸<http://www.myri.com/scs/GM-2/doc/html/>

plane for passing control information like token for FCT protocol. According to manufacturer’s specifications and benchmarks, it features as low one-way latency as $6.3 \mu\text{s}$ for short messages up to approximately 100 B with MPI and GM-2.0 API⁹.

- $6 \times$ PCs used as the parallel nodes for the distributed AE with configuration shown in Table 1. Each node was connected to via data link to HP switch via full-duplex Gigabit Ethernet and also to Myrinet switch for control information passing.
- Sender and receiver PCs with the same configuration (Table 1), that have been used for generating traffic and collecting and analyzing results. Both computers were connected to the HP switch via full-duplex Gigabit-Ethernet.

<i>Configuration</i>	
Brand	HP ProLiant
Model	DL 360 G3
Processor	$2 \times$ Intel Xeon 2.40 GHz
Front-side bus	533 MHz
Memory	2 GB (PC 2100 DDR)
GE NIC	$2 \times$ Broadcom Corporation NetXtreme BCM5703 (rev. 2)
Myrinet NIC	M3F-PCI64C-2
Operating system	Linux Debian Woody kernel 2.4.29 SMP
GM Socket version	2.0.8

Table 1: Configuration of distributed AE nodes and sending/receiving probes.

The data flows were generated using simple RTP-compliant sending application and data reception was done by receiver, which also computed all the required statistics. We have evaluated it with sending data stream with bandwidth up to 1 Gbps and measurements above 1 Gbps will be carried out in the near future.

3.2 Performance Evaluation

Performance of the distributed AE prototype without explicit sending synchronization and with FCT-based synchronization is shown in Figures 7 and 8 respectively.

It turns out that single path AE (equivalent to single reflector) is not capable of processing streams beyond 600 Mbps on given testbed infrastructure without packet loss. This is in accordance with findings in [15], where stand-alone centralistic reflector was examined on testbed infrastructure with even slightly less performance.

Contrary to the stand-alone reflector, the distributed AE prototype can process and distribute streams up to 1 Gbps using 1472 B UDP payload without significant packet loss starting with two parallel paths. Jitter, usually explained as delay dispersion, is calculated according

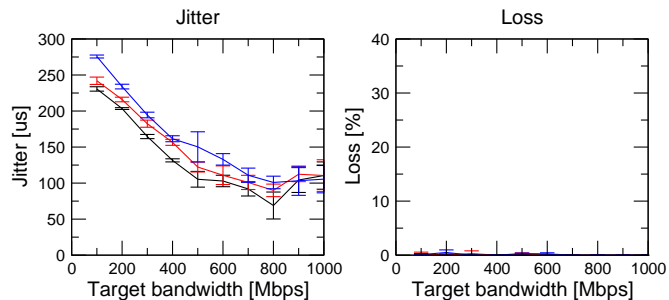
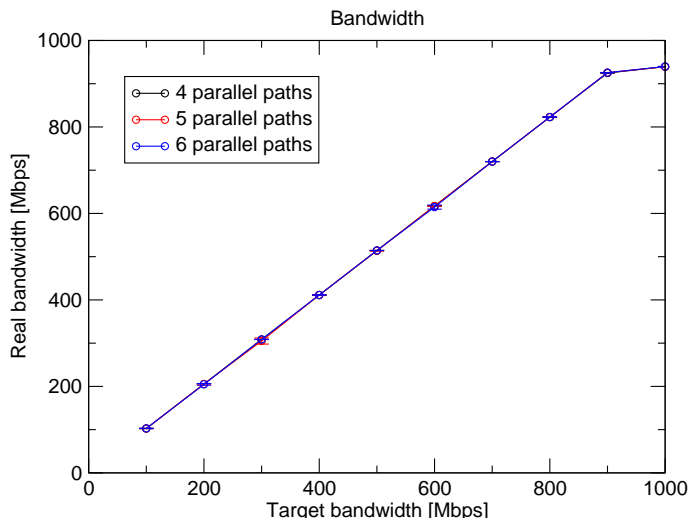
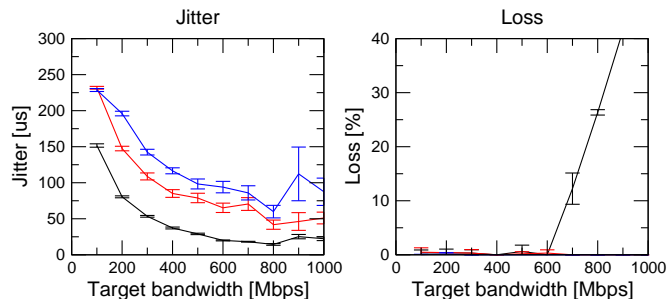
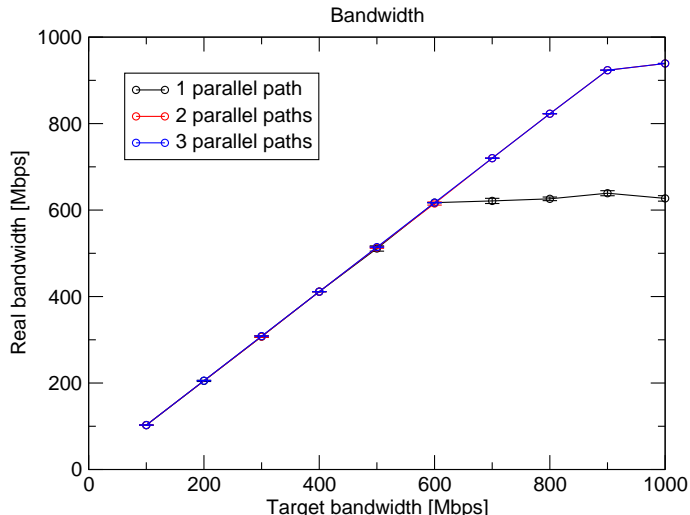


Figure 7: Forwarding performance of distributed AE without explicit synchronization for number of paths 1 through 6.

⁹<http://www.myri.com/myrinet/performance/#GM-2.0>

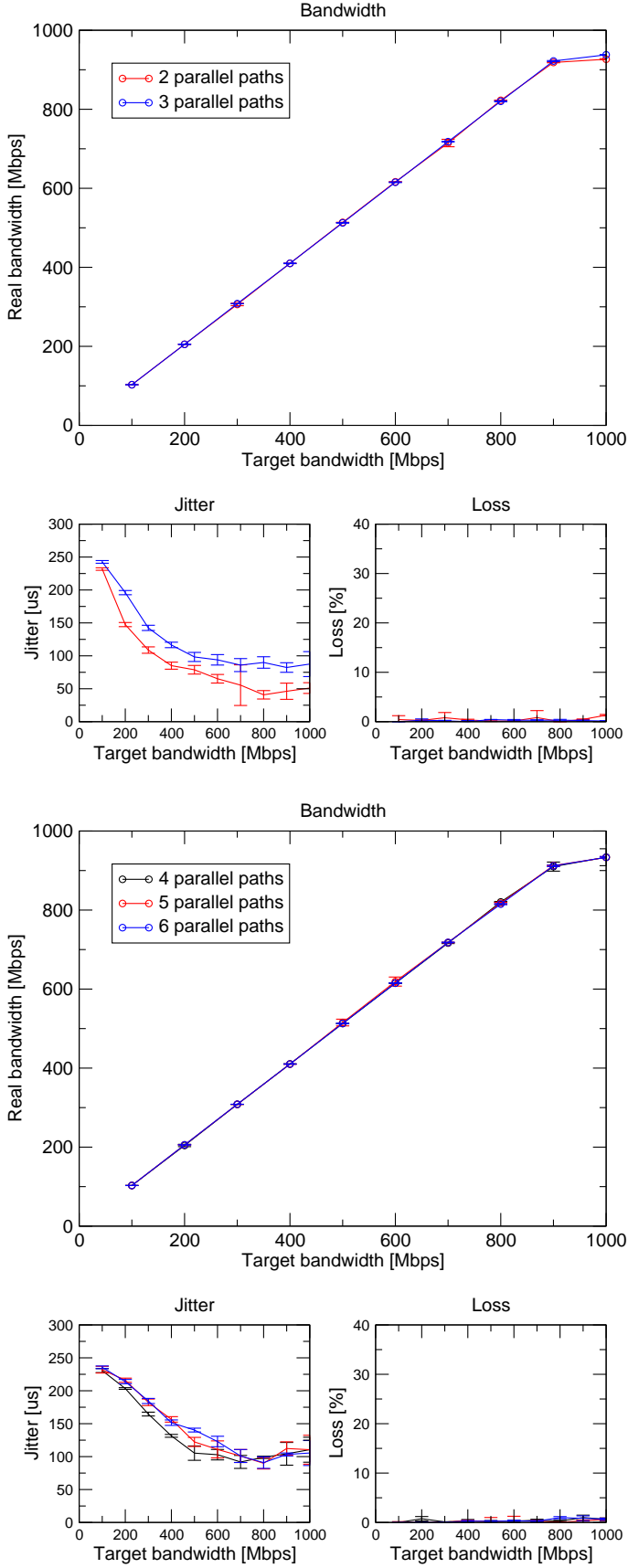


Figure 8: Forwarding performance of distributed AE with synchronization using FCT for number of paths 2 through 6.

to RFC 3551 based on arrivals of two consecutive packets as

$$D_{i,j} = (R_j - R_i) - (S_j - S_i) = (R_j - S_j) - (R_i - S_i),$$

$$J = J + \frac{1}{16}(|D_{i-1,i}| - J),$$

where S_i is the RTP timestamp from packet i , and R_i is the time of arrival in RTP timestamp units for packet i for two consecutive packets i and j . From this definition it doesn't include packet reordering discussed below and it only measures "evenness" of packet arrivals independent of packet order. It starts about $300 \mu\text{s}$ when sending 100 Mbps and slowly drops to below $100 \mu\text{s}$ as the data rate increases. For more parallel paths, it slightly raises and this effect is more pronounced on distributed AE without egress synchronization.

3.3 Packet Loss and Reordering Evaluation

We have measured and analyzed also packet reordering in order to experimentally compare behavior of distributed AE without explicit egress synchronization and distributed AE with synchronization using FCT protocol. The detailed reordering samples can be found in [14].

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 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 and $\max\{j\}$ is the maximum positive difference.

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

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

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:

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

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 [14]. 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.

With no egress synchronization				With FCT protocol			
<i>2 parallel paths</i>							
<i>BW</i> [Mbps]	$\min\{j\}$	H^-	N^-	<i>BW</i> [Mbps]	$\min\{j\}$	H^-	N^-
100	-15	-58	18	100	-1	-19	19
200	-7	-35	11	200	-3	-242	184
300	-49	-339	55	300	-3	-502	466
400	-13	-194	150	400	-3	-4768	4724
500	-15	-1370	1266	500	-3	-4563	4533
600	-19	-12240	11726	600	-3	-36270	36220
700	-25	-48405	47871	700	-3	-109264	109238
800	-33	-105801	103793	800	-3	-176277	176269
900	-35	-239722	234334	900	-3	-98721	98703
1000	-35	-265286	258558	1000	-3	-55548	55476
<i>3 parallel paths</i>							
<i>BW</i> [Mbps]	$\min\{j\}$	H^-	N^-	<i>BW</i> [Mbps]	$\min\{j\}$	H^-	N^-
100	-13	-37	11	100	-2	-16	13
200	-32	-149	20	200	-5	-269	154
300	-376	-28171	1230	300	-5	-606	432
400	-113	-2452	316	400	-5	-1357	976
500	-17	-576	474	500	-5	-2169	1897
600	-104	-3759	1473	600	-5	-5688	5268
700	-154	-9608	4129	700	-5	-9180	8303
800	-26	-10147	8821	800	-5	-14522	13069
900	-32	-25210	20327	900	-5	-32276	28228
1000	-32	-31032	24806	1000	-5	-29819	25872

Table 2: Comparison of reordering for distributed AEs with no explicit egress synchronization and with synchronization using FCT. Part 1.

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 $j = -1$.

Stand-alone reflector. As discussed above, the stand-alone reflector is not capable of forwarding data streams above 600 Mbps without packet loss, where only +1 reordering value is only populated up to 600 Mbps and asymmetrical distribution leaning toward positive values is shown for 700 Mbps and above. The bigger the loss is, the larger the sum H^+ is. Because no reordering nor duplicates are introduced either, $H^- = N^- = h_0 = 0$.

Distributed AE without explicit synchronization.

While the distributed AE performs very well in terms of low packet loss, it introduces severe packet reordering in both terms of maximum reordering (expressed as the minimum reordering populated in histogram, i.e. $\min\{j\}$) and also numbers of packets reordered (expressed by H^- and N^-), as obvious from left column of Tables 2 through 4. Furthermore, the reordering fluctuates very significantly in time and is hardly reproducible.

Distributed AE with synchronization using FCT protocol.

Compared to distributed AE without explicit synchronization, the FCT protocol allows distributed AE to work in much more predictable manner. It reduces packet reordering to just a very few packets and it also reduces reordering both in terms of maximum reordering ($\min\{j\}$) and numbers of packets reordered (H^- and N^-), as can be seen from right column of Tables 2 through 4. The maximum reordering grows $\min\{j\} = 2n + 1$ where n is number of parallel AE paths, which indicates that the switch in the testbed works in rather very precise round robin fashion when aggregating flows from multiple interfaces. The number of reordered packets is comparable for lower number of parallel paths for both with and without synchronization and as the number of parallel path grows, the synchronized version becomes better more than $3\times$ for higher bandwidths. The results are also in agreement with the theoretical analysis of *maximum* reordering, as $n(s_i^{SW} + 3) > 2n - 1$.

Detailed reordering histograms for both distributed AE without explicit synchronization and with it can be found in [14].

Token round-trip time has been also periodically sampled¹⁰ and it ranges between 14 μ s for 2 parallel AE paths

¹⁰In order not to influence results of measurements of distributed AE performance, it was not possible to continuously gather individual token round-trip times. Thus only sample values were periodically gathered.

<i>With no egress synchronization</i>				<i>With FCT protocol</i>			
<i>4 parallel paths</i>							
<i>BW</i> [Mbps]	<i>min{j}</i>	<i>H⁻</i>	<i>N⁻</i>	<i>BW</i> [Mbps]	<i>min{j}</i>	<i>H⁻</i>	<i>N⁻</i>
100	-14	-63	17	100	-2	-12	11
200	-41	-194	86	200	-5	-85	74
300	-35	-6632	6546	300	-7	-491	298
400	-13	-141793	141701	400	-7	-4897	4526
500	-15	-499869	499478	500	-7	-13894	13214
600	-106	-543649	535408	600	-7	-94937	91991
700	-119	-578858	542561	700	-7	-254382	243885
800	-38	-683344	547351	800	-7	-375694	359466
900	-31	-976225	552171	900	-7	-466649	418082
1000	-30	-1022286	553458	1000	-7	-482681	425897
<i>5 parallel paths</i>							
<i>BW</i> [Mbps]	<i>min{j}</i>	<i>H⁻</i>	<i>N⁻</i>	<i>BW</i> [Mbps]	<i>min{j}</i>	<i>H⁻</i>	<i>N⁻</i>
100	-2	-17	13	100	-2	-18	17
200	-6	-98	61	200	-4	-139	115
300	-21	-3683	3584	300	-9	-729	574
400	-11	-159904	159808	400	-9	-2299	1575
500	-44	-418933	417555	500	-9	-18685	17901
600	-22	-434274	425202	600	-9	-40039	37833
700	-156	-493158	435529	700	-9	-121278	116791
800	-42	-852873	519740	800	-9	-288200	276433
900	-28	-1458732	736498	900	-9	-382566	349666
1000	-152	-1520055	762531	1000	-9	-380204	349372

Table 3: Comparison of reordering for distributed AEs with no explicit egress synchronization and with synchronization using FCT. Part 2.

and raises up to $40\ \mu\text{s}$ for 6 parallel paths. This closely approaches manufacturer-stated one-way message passing latency of Myrinet configuration used for the testbed as described above.

4 Conclusions

When parallelizing individual AE itself, the distributed AE comprises multiple equivalent AE units running in parallel and data distribution unit, which distributes data over the multiple parallel paths in the distributed AE, and data aggregation unit, which aggregates resulting data from the parallel paths into one or more output network links.

Distributed AE brings a new problem inherent to the fact that the data is flowing through multiple independent paths—packet reordering. It results either into packet loss (if delayed out-of-order packets are just discarded) or indirectly into latency increase as the application needs to buffer the data in order sort packets before actual processing begins. For cases where better than no explicit sending synchronization is needed to minimize output packet reordering induced by the distributed AE, we have designed and evaluated Fast Circulating Token protocol providing limited synchronization among sender modules of distributed AE parallel paths. While distributed AE with no explicit sending synchronization provides limited reordering, we have shown both theoretically and experimentally that FCT decreases maximum

egress packet reordering, sometimes even more than two orders of magnitude.

Regarding future work, the distributed AE could be complemented with programmable hardware implementation of load distribution and especially aggregation, which could be used for implementing sending in exact order, yielding zero packet reordering. The implementation could be based on FPGA-based programmable network interface cards with multiple interfaces.

Furthermore we would like to study different audio and video processing algorithms suitable for implementation on distributed AE. Such algorithms must minimize state sharing in order to provide efficient and scalable processing.

Finally, we would like to design an AE with built-in Quality of Service on all levels, including underlying operating system, for strict separation of different processors as well as different flows. Actually, many multicast deployment related problems, which initiated our research into overlay networking, could be mitigated if the multicast-enabled routers had strict separation of different processes running inside. In the longer term we would like to design and prototype a distributed router with such QoS capabilities.

Acknowledgments

This project has been supported by a research intent “Optical Network of National Research and Its New Ap-

With no egress synchronization				With FCT protocol			
6 parallel paths							
BW [Mbps]	min{j}	H ⁻	N ⁻	BW [Mbps]	min{j}	H ⁻	N ⁻
100	-16	-80	17	100	-3	-18	13
200	-7	-78	47	200	-5	-49	36
300	-47	-1955	1804	300	-9	-418	359
400	-62	-62744	62452	400	-11	-2596	1823
500	-14	-331120	330329	500	-11	-7553	6638
600	-17	-371489	357821	600	-11	-27938	25856
700	-124	-397293	363594	700	-11	-140596	135701
800	-27	-517597	379041	800	-11	-171797	164183
900	-31	-1252132	629255	900	-11	-324345	297198
1000	-25	-1237362	622667	1000	-11	-330780	302301

Table 4: Comparison of reordering for distributed AEs with no explicit egress synchronization and with synchronization using FCT. Part 3.

plications” (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] Petr Holub, Eva Hladká, and Luděk 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*, volume 3421/2005 of *Lecture Notes in Computer Science*, pages 876–883, La Réunion, France, April 2005. Springer-Verlag Heidelberg.
- [2] Petr Holub, Eva Hladká, Jiří Denemark, and Tomáš Rebok. Active elements for high-definition video distribution. In *ICT 2006, 13th International Conference on Telecommunications*.
- [3] Philippe Galvez. From VRVS to EVO (Enabling Virtual Organizations). In *TERENA Networking Conference 2006*, Catania, Italy, May 2006.
- [4] Markus Buchhorn. Designing a multi-channel-video campus delivery and archive service. In *The 7th Annual SURA/ViDe Conference*, Atlanta, GA, USA, March 2005.
- [5] Tzi-cker Chiueh and Prashant Pradhan. Suez: A cluster-based scalable real-time packet router. In *The 20th International Conference on Distributed Computing Systems (ICDCS 2000)*, Taipei, Taiwan, April 2000.
- [6] Tzi-cker Chiueh and Prashant Pradhan. Suez: A cluster-based scalable real-time packet router. Technical Report TR-65, Experimental Computer Systems Lab, State University of New York, April 2000.
- [7] D. Decasper, G. Parulkar, and B. Plattner. A scalable, high performance active network node. *IEEE Network*, 33(1):8–19, January 1999.
- [8] P. Graham. A DSM cluster architecture supporting aggressive computation in active networks. In *Intl. Workshop on Distributed Shared Memory*, 2001.
- [9] Young Bae Jang and Jung Wan Cho. A cluster-based router architecture for massive and various computations in active networks. In *ICOIN*, KAIST, Korea, February 2003.
- [10] M. Aron, D. Sanders, P. Druschel, and W. Zwaenepoel. Scalable content-aware request distribution in cluster-based network servers. In *Proc. USENIX Ann. Technical Conf.*, June 2000.
- [11] R. Bianchini and E. V. Carrera. Analytical and experimental evaluation of cluster-based www servers. *World Wide Web J.*, 3(4):215–229, December 2000.
- [12] 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*, pages 205–216, October 1998.
- [13] Enrique V. Carrera and Ricardo Bianchini. Press: A clustered server based on user-level communication. *IEEE Transactions on Parallel and Distributed Systems*, 16(5):385–395, May 2005.
- [14] Petr Holub. *Network and Grid Support for Multimedia Distribution and Processing*. PhD thesis, Faculty of Informatics, Masaryk University Brno, Czech Republic, 2005.
- [15] Eva Hladká. *User Empowered Collaborative Environment: Active Network Support*. PhD thesis, Faculty of Informatics, Masaryk University Brno, Czech Republic, 2004.