High Speed Network Monitoring

Leonhard Uden
Betreuer: Nathan Evans
Seminar Future Internet SS2010
Lehrstuhl Netzarchitekturen und Netzdienste
Fakultät für Informatik, Technische Universität München
Email: uden@in.tum.de

ABSTRACT

Network monitoring is an important activity to ensure a smooth working of IP networks. Operators of networks utilize it to measure the traffic, plan new investments or perform intrusion detection. Since the bandwidth of network links increases much faster than the processing power, centralized architectures are no longer capable to capture and monitor the amount of traffic or depend on high performance hardware.

Distributed architectures which split up the traffic to several capturing nodes are a promising solution to this problem. They provide scalability and consist of modified standard PCs. This paper will present a proposal of such an architecture and compare its design to other approaches. Different dispatch methods will be compared and analyzed. The performance of distributed designs outperforms centralized ones when they use similar hardware. Issues still exist, since load balancing and comprehensive monitoring is difficult to achieve at the same time.

For now centralized architectures using high performance hardware are the better solution for network operators. Web development and different growth rates of bandwidth and processing power lead the development of high speed monitoring towards a distributed solution.

Keywords

Network Monitoring, Distributed Architecture, High Speed Networks.

1. INTRODUCTION

In recent times the amount of data transferred by IP networks constantly increased. This has happened by providing new services which are often free of cost. Older analog technologies are getting displaced by newer ones using IP networking. Skype and youtube are the most famous examples of this trend.

In order to guarantee a certain quality of service, detect malicious traffic and plan new investments, network operators have to monitor and analyze traffic. They are challenged by this development, since the growth in amount of data transferred is much higher than the growth of processing and memory speed. A forecast of Cisco expects the IP traffic to be five times higher in 2013 than in 2008, which means a yearly growth rate of 40 percent [2]. When it comes to IDS (Intrusion Detection Systems) the increasing possibilities of malicious traffic extend the process of scanning a packet. Nowadays it is common to use 10 Gigabits per second Ethernet links in bigger networks. Such a fully loaded link transfers a packet in less than 100 nanoseconds, by decreasing packet-sizes, therefore increasing packet-number, the processing time for each packet decreases. That means a capturing device has only nanoseconds to capture a packet. Also the time needed for analyzing packets has to be considered. There are different ways to deal with this challenge.

The most obvious one would be to use special high performance capturing devices which are able to capture at Gigabits per second Ethernet transfer rates. The dedicated capturing cards of Endance[4] provide capturing at high transfer rates. These cards guarantee to capture 100 percent of the packets, are able to distribute the traffic to different memory buffers, perform time stamping and reconstruct a replication of traffic. Drawbacks are higher prices and less flexibility compared to other approaches.

The next option is to lower the capture rate by using sampling. This method is popular today because it is sometimes already implemented into routers and does not have a need for high performance hardware. Here the questions are how high the sampling rate should be and if one can make founded conclusions to the rest of the packets which are not captured. Research has shown that sampling is accurate in computing the total amount of traffic, since the packet-size does not vary too much. Sampling has a lack of accuracy when it comes to flow count, smaller flows could be missed entirely. That often causes anomalies to stay undetected [1]. Recent research on sampling methods has found ways to increase the rate of detection, but this won't be handled in this paper, overviews are given in [14].

This paper will describe distributed architectures. These architectures try to split the traffic and allocate it to different capturing or analyzing devices, in order to achieve a high capture rate and avoid sampling. There have been several projects on this topic recently, like the IDS architecture described in [12] or the DaSahit project (Distributed and Scalable Architecture for High-Speed IP Traffic Analysis) [3]. Different rules are applied to split the traffic: to sort the packets by port, IP address, or by using some sort of algorithm.

DiCAP(Distributed Packet Capturing Architecture for High-Speed Network Links) [9] will be used as an example for such an architecture. It is especially designed to capture traffic, not to analyze the packets, which means it only retrieves the data of a network link and does not scan the data. The aim is a scalable architecture that can be flexibly extended. A major advantage of this method is the possibility to split the traffic into such small amounts that it can be processed by inexpensive standard PCs using Linux.

The paper is organized as follows. In section 2 DiCAP will be presented and reviewed. Section 3 compares different approaches of creating a distributed architecture for monitoring or analyzing IP traffic. The last section, number 4, concludes the findings of this paper.

2. DiCAP

Uncoupling packet capturing from its limitations is a major motivation for distributed architectures. The authors of DiCAP present a proposal that is able to handle all kinds of protocols, is not dependent on high performance hardware neither lacks accuracy like centralized sampling methods. It is an easy scalable architecture using standard hardware.

Distributing the workload of capturing packets of high speed links to multiple devices is the main idea of distributed architectures. In DiCAP those devices are called capture nodes, are based on regular PCs with a modified NIC driver. They are connected with a router called mirroring device, a standard PC which coordinates them and eventually with an external analyzing device which is not considered in this paper. The distribution of workload is achieved by forwarding the traffic to each capture node and using a kind of sampling on each that only allocates a unique selection of the packets to each node. How this is done will be explained and analyzed in the next sections, which are all based on the paper written by Morariu and Stiller [9].

2.1 Architecture

On a network link a mirroring device is installed, its task is to mirror the passing traffic and to forward the copied packets on to the capture nodes. A drawback is that if mirroring fails, capturing and analyzing fails. Security dependent networks might also use the mirroring device as a “gate” on the main link, so if the device fails, mirroring is no longer possible, but also no traffic is able to pass. Via multicast the packets are sent to the capture nodes.

The cluster of capture nodes is organized by a node coordinator. It tells each capture node how to decide which packets to capture and which to discard. To avoid a breakdown of the whole system, because of a malfunction of the node coordinator, other node coordinators could be installed. They would be synchronized every n seconds and stay inactive, until the active coordinator fails and another one is chosen as a replacement.

Every capture node has a unique nodeID that is used for identification when the topology is set at the node coordinator. A capture node has two interfaces, a passive one just for retrieving the mirrored traffic and an active one for communicating with the current active node coordinator and the packet data analyzer(s). When a packet arrives at the capture node, the DiCAP module decides whether to drop the packet or to capture it. The DiCAP module is a software extension of the NIC driver (see Section 2.1.3). If the packets should be captured is decided by using a function. The authors propose the use of a hash based selection or round robin selection. Which one to be used is defined by the node coordinator.

There are two different solutions existing. The first one is called distributed capture mode (see Figure 1). In this mode the DiCAP module stores only the first 54 bytes in a buffer, so only the packet headers are stored. The payload is dropped, this leads to a lack of data integrity. If the buffer is full a UDP message is sent to the packet data analyzer(s).

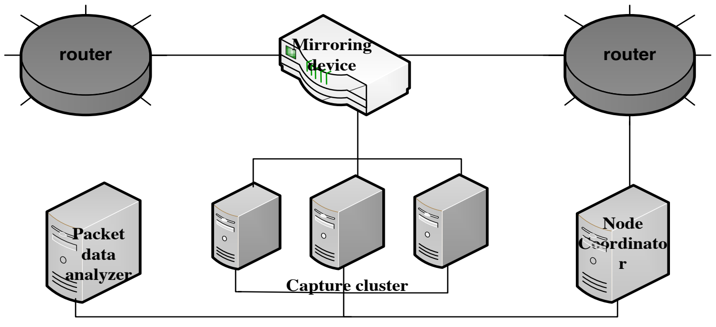

Figure 1: Distributed Capture Mode

How the captured packets are distributed may be discussed in later sections, since this is only a description of the monitoring architecture.

The other mode is called distribution mode (see Figure 2). Here the DiCAP module is not capturing the packets, but forwards the packets it decided to keep to an external capture tool. The prototype uses libpcap[7] for this task. So DiCAP is only a distributor in this mode.

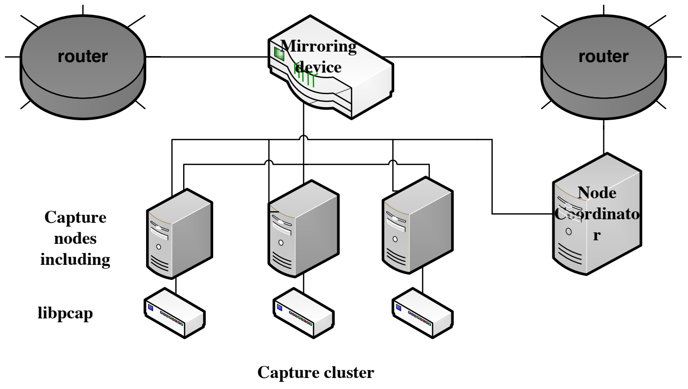

Figure 2: Distribution Mode

2.1.1 Communication

In order to achieve correct traffic distribution without sampling effects, it is necessary that every capture node is aware which part of the traffic it should capture. Since the distribution can change when new nodes enter or leave the cluster, the node coordinator has to be informed about changes and update the topology regularly. A node is an active member as long as it regularly sends heartbeat messages.

This process is handled by sending control messages via the active interface of the nodes. The generic control messages are specially defined for this task. They use seven different so called AVPs (Attribute-Value-Pairs) to exchange information between the coordinator and the capture nodes. The

- nodeID which identifies the capture node
- Coordinator IP/Port which tells the capture nodes which IP address/port is used by the coordinator
- Analyzer IP/Port which tells the capture nodes where they should forward the captured data
- Selection Type which tells the capture nodes how to select the packets to be captured

- Validity Start which specifies when new topologies should be coming into effect

A selection of these values is sent with the different messages exchanged. The three message types are called join, accept and topology update. In a join message a node who wants to join the cluster sends a message with his nodeID to the coordinator. An accept message confirms the joining. By sending a topology update message the coordinator defines the topology of the cluster.

2.1.2 Dispatch Method

As mentioned before the authors chose two different distribution solutions. One of them is a round robin selection mode, in this mode every capture node has a definite position (P_a) in the total amount of active capture nodes (N). Each node has a packet counter C, which is always set to zero when a topology update happens. A node captures a packet if:

$$C \bmod N = P_a$$

else the packets are dropped. This selection mode allocates every capture node a data amount of 1/N of the total data, which means a perfectly balanced workload. This way each capture node processes sampling but all together they capture the whole traffic. The mode only works, if a perfect synchronization of the individual nodes is given. The different counters always have to be equal and the traffic received always in the same order, else traffic will be missed or captured twice. All nodes have to get all update messages and work always properly to ensure an exhaustive capturing process.

Another method utilizes a hash function. The challenge is to find an appropriate hash function which is easy to calculate and has well balanced outcomes. Also a well distributed value as base for input for the hash function is needed. With the position of the capture node (P_a), the total number of active nodes (N) and an input value (I) given, packets are captured if:

$$\text{hash}(I) \bmod N = P_a$$

else the packets are dropped. The authors discovered that the identification field in the IP header could meet the expectations of a well balanced input value. A header field that identifies the fragments of a packet and is used for reassembling does not include information about flow membership, additionally in IPv6 the field is only existent in an extension header. To use a header field in combination with a hash function might be a good solution, but it is hard to find an appropriate header field and a hash function which meets the criteria of load balancing and flow preserving.

If the main criteria for monitoring, is just to measure the total amount of packets and perfect load balancing, the round robin selection mode should be preferred. It provides a perfectly balanced distribution of traffic and incrementing a counter is a cheap operation. It should not be a problem to keep the traffic in order since only short Ethernet links are used. A concern of the round robin selection mode is the synchronization of the individual counters. If it is possible to synchronize the individual capture nodes when a topology update occurs is uncertain. If a node captures packets of a 10 Gbps network link, it might has to capture a packet every 40 ns, therefore the counter of the nodes have to be accurate to nanoseconds. If the counters are not synchronous, different nodes capture the same packets and therefore some packets will not be captured because they are not in the scope of any node. A solution to this problem might be to interrupt the traffic forwarded by the router for a short period of time, in order that the capture nodes have more time to reset their counters. Only if the synchronization is ensured, the total amount of packets is captured. Topology updates will not happen very often once the architecture is established, therefore the interruptions of the mirrored traffic are negligible.

For comprehensive monitoring it is necessary that flows are detected and captured at the same device. This is important for flow path analysis and IDSs. It is not reasonable to split flows and try to reassemble them later on. That would cause an expensive processes and very good communication between the different nodes is necessary. A large distribution process to exchange packets between the capturing devices and analysis devices would be the outcome. A method which sorts the packets by their attributes in the header, eventually combined with a hash function, is necessary for an effective comprehensive monitoring.

2.1.3 Capture Node

Every capture node could be a standard PC, the only specification is that it has two network interfaces. Capturing is organized by a so called DiCAP module that is also implemented in the coordinator nodes and the packet data analyzer (if they exist). This module is a configuration of the driver of the NIC (Network Interface Card). Its task is to decide whether to capture a packet, forward it or drop it, before the kernel allocates memory to the packet. The DiCAP module consists of a management unit, a packet processor and a packet data forwarder (see Figure 3).

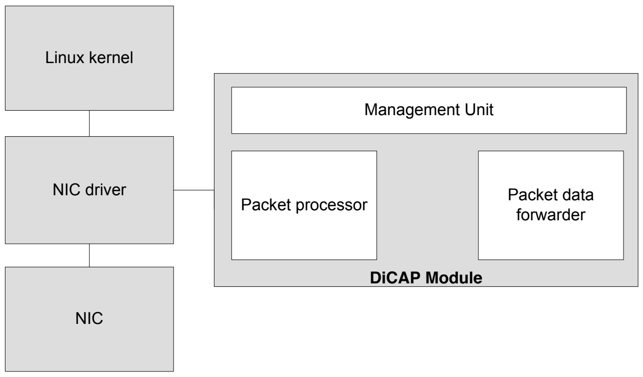

Figure 3: DiCAP Module

Communication with the node coordinator and determining the behavior of the module is the task of the management unit. It has information about the addresses and which network topology is used. The packet processor only handles packets received in the passive interface, the others are sent to the kernel. It has a different behavior for each mode. In distributed capture mode the packets received on the passive interface are sent to the packet forwarder if they are in the responsibility of the capture node, if not they are just dropped. The monitored packets are never processed by the kernel in this mode and always dropped by the NIC driver. Packets to be monitored are delivered to the packet forwarder, which stores the headers in a buffer that it sends to a packet analyzer via UDP when the buffer is full.

While using the distribution mode, the packet forwarder is deactivated, since the captured packets are locally processed by libpcap. Here the packets that should be captured are forwarded to the kernel and other packets are dropped by the NIC driver. In this mode the task of the DiCAP module is not to capture the packets but to determine which ones to analyze at which analyzing device.

2.2 Review

As seen for the purpose of just capturing packets at high rates DiCAP is a well balanced architecture. But if the project should ever be used further there are some criteria that hinder use in a more complicated environment.

The implementation of the design is good when it comes to scalability. To extend the capture cluster a pretty automatic procedure was created. With the creation of a new protocol to communicate between the coordinator and the capture nodes it is easy handled for a node to join the cluster. One join message, one accept message and a topology update suffices to extend the cluster. Another quality is the opportunity of combining the system with other software, as is done with libpcap in the distribution mode.

Fault tolerance is achieved by the system of multiple coordinator nodes and heartbeat-messaging of the capture nodes. If a capture node does not send its heartbeat message for a determined time, the node coordinator erases it from the list of active nodes and performs a topology update automatically. If a node coordinator breaks down it is replaced by another one which was synchronized before. The weakness is the mirroring device. It cannot be easily replaced and only exists once in the prototype.

Performance tests by the authors have shown that a single DiCAP device, used on a single node in distributed capture mode where only the first 54 bytes of each packet are captured, outperforms other devices using libpcap or libpcap-PFRING [11]. At high packet rates of 620Kpps(thousand packets per second) where the libpcap and libpcap-PFRING devices have a loss rate of 93 percent and 96 percent, the DiCAP device still has a packet loss rate of 0 percent. When used in distribution mode in combination with libpcap a performance increase by increasing number of nodes is remarkable. It is also reasonable to use two different devices for communication and monitoring, since communication packets always have to be processed and some monitoring packets are dropped.

The implementation of the DiCAP module prevents the kernel from allocating memory to packets which are not captured. Since first the module decides whether to drop a packet or not and second memory is allocated to packets which should be processed by the kernel.

Round Robin distribution results in a perfect load balance, but when the packets are further processed it might be useful to capture comprehensive flows. If round robin is used like this, the packets are captured with no relation to each other. Using the identification field of the IP header as an input for a hash function might also result in a balanced load, but to find a function that allocates all packets of a flow to one node and balances the load on all nodes might be very hard to find.

Altogether DiCAP is a well designed architecture, where the authors spent time on thinking about good solutions for scalability, fault tolerance and performance. But to be useful for network operators who want to monitor packets in order to analyze them, another distribution method has to be found.

3. Comparison of different architectures

In order to get an overview of the different forms of distributed architectures, recent proposals of other scientists are presented and compared in the way they handle the upcoming problems of such a solution. It has to be said that comparison is somehow difficult because technology made such big steps over the years. Some implementations handle 100 Mbit network links, others 10 Gbit links. Also the different architectures are often designed for different purposes. The next sections will compare the design, the way packets are distributed and how the capture nodes work.

3.1 Architecture

All studied architectures have in common that they are designed for high speed network links. Normally the packets are copied by a mirroring device or an Ethernet switch. An anomaly of DiCAP is the lack of an active distribution device. Most other designs include an active distributer. For this task a router or an Ethernet switch is neccessary. The mirroring device of DiCAP has a similar task, but it does not decide which capture node gets which packet, the traffic is just forwarded to all capture nodes. To spread the selection process to the single capturing nodes which execute the given rules, like DiCAP does, might help to avoid a bottleneck at the distribution device.

DiCAP uses a coordinator node to organize the capture cluster, other approaches mainly miss a coordinating device. In [6] a manager device is used to advertise definitions for analyzing packets, collect reports and add or remove capturing devices. Flexibly adding and removing capture nodes, like it is possible in DiCAP is not possible in other architectures like [12]. Since they mostly use capturing devices explicitly defined for a special scope.

With the exception of DiCAP, almost all examined approaches combine the capture device and the analyzing device. This leads to a higher processor load of the individual nodes but avoids the need of transferring captured data to an analyzing device and an eventually required distribution process. This design decision is a question of processing power and the number of analyzing nodes, thus a question of traffic per node. An external storage center as proposed in [8] will be necessary when capturing traffic on high speed links for longer periods of time.

3.1.1 Dispatch Method

The critical concern of the distributed designs is the question of how to dispatch the traffic. There are some major principles the method should fulfill:

- In order to minimize packet loss and regarding the processing power limitations, load balancing has to be achieved.
- To draw comprehensive conclusions of the captured packets, the dispatch process has to consider the logical memberships of the packets, such as flows, source or protocols used.

As already presented above, one method is to use a round robin distribution method that simply spreads the traffic into even parts by allocating every n-th node the mod n-th packet. It is a cheap process and may be sufficient for the first criteria mentioned. But it is not suitable for the second one. A solution might be to add an analyzing device that creates a kind of reassembly list which enables reconstruction of sessions for example [12]. Since every packet has to be identified, a big overhead would be the result.

Most other approaches categorize the packets by one of their attributes and not by the order they are sent. Splitting the traffic by their destination port is an often realized suggestion, since it fulfills the second criteria. When packets are dispatched like this, it is for example possible to reconstruct TCP sessions or to analyze a flow path. Load balancing is hardly given. Since there are far more HTTP packets than FTP packets transmitted there is likely to be an imbalance between the workload of the different capturing devices. A solution is to allocate different numbers of ports to each node, so that the estimated load is equal for every capturing node. Even one port could be divided between two nodes to capture. The authors of [8] developed a system that sorts the packets by their IP source addresses and allocates them to different nodes by using a greedy algorithm. Estimated traffic for each IP source address segment is appraised and every node gets a similar amount of estimated traffic allocated. If the traffic is monitored longer the load balance is almost existent. A promising way to reduce load per node is to use different stages of processing, so two packet attributes could be used to dispatch the traffic. First the packets could be split by their port destination and in a second layer packets could be dispatched by their source IP address (see Figure 4). This method ensures the second criteria, since a flow is identified by its IP source\destination address, the source\destination port and a layer four protocol. So packets within the same flow will be captured at the same node. Also great potential for load balancing exists because the traffic can be divided and load balanced on two layers. The authors of [5] are using this method to analyze packets.

Another suggestion is to assign an identifier to every flow and every node a field of responsibility which flows to capture [10]. This way the captured packets could be distributed evenly by using a round robin algorithm and the packets of one flow could be all captured at the same node.

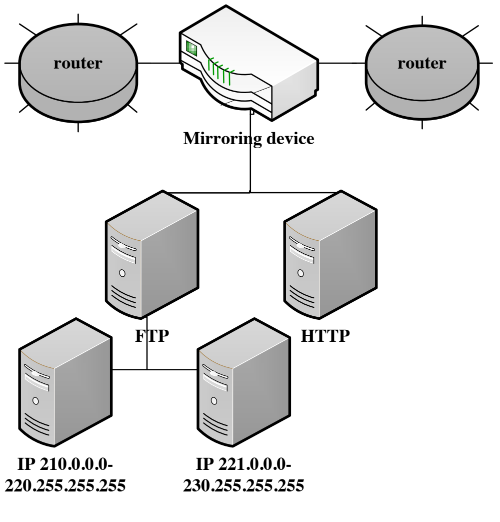

Figure 4: two layer distribution

A problem with attribute dependent dispatch methods is the susceptibility to denial of service attacks. For example, if the network is flooded with packets using just one port. In contrast to round robin methods the attribute filtering is inflexible towards increasing traffic. Round robin methods, like the one used in DiCAP, distribute additional packets always evenly between the different nodes. Another drawback of the attribute filtering methods is the low expandability compared with round robin. New capturing nodes cannot be just added, since every new node has to be assigned to a special attribute. Only the existing nodes which use the same attribute as the new node will be relieved. In round robin every capture node is relieved, when a new node is added.

3.1.2 Capturing Device

In order to achieve scalability, the capture devices are normally standard PCs providing several interfaces. In some proposals the capture nodes also serve as an analyzing device, others forward the captured data to an external analyzer. The module to decide whether to capture a packet or not should be implemented as low as possible in the capturing device [9] in order to avoid wasting processing power on useless packets. Most drafts are based on Linux kernels and use a software tool like libpcap[7] to capture packets.

3.2 Performance

As mentioned before a comparison of the different proposals is hard because of different hardware and software used and different composition and amount of traffic monitored. No independent evaluation was possible, so all numbers are based on the data of the different authors. This paper mainly presents the evaluation results of several architectures presented.

The authors of [8] show that their implementation of a distributed architecture, using a round robin method to dispatch the traffic, experiences no packet loss using four capture nodes, where a single centralized capture device loses 90 percent of the packets. An IDS architecture [12] using seven nodes, a dispatch method dependent on the destination ports of the packets and an optimized number of Snort [13] rules for each capture/analyzing node. This method reduces the number of snort rules on each node, by checking only the Snort rules which are relevant for each port. This method performs up to ten times faster than a single centralized device. The authors of [6] show that a division of flow comparison patterns increases the packet capture rate. Evaluation of DiCAP has shown that libpcap on one node is not able to capture every packet. By using the distribution mode of DiCAP and round robin selection, the packet capture rate increases by the number of nodes. Figure 5 shows the packet capture rates for different numbers of nodes at different rates of packets per second. At some rates four nodes capture ten times more packets than one node.

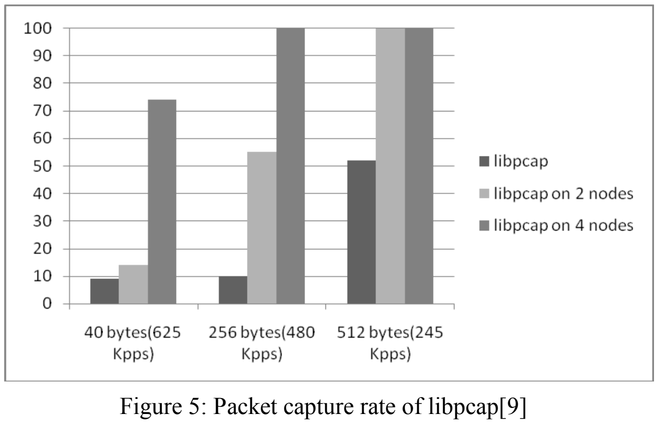

Figure 5: Packet capture rate of libpcap[9]

The results demonstrate that packet loss rates or time for analyzing packets is decreased by taking advantage of multiple capture devices and splitting up the rules to analyze packets. Distributed architectures always outperform centralized ones, when the same hardware is used. It has to be considered that distributed architectures consist of several capture or analyzing devices. Several devices provide more processing power than a single device. The weakness of many evaluations studied is that they only compare their architectures to single devices. A real boost in performance is only existent if the architecture does not only distribute the workload, but also optimize the processing of the workload. An example for such architecture is presented in [12].

4. CONCLUSION

The reviews of the different proposals made on distributed architectures designed for high speed networking monitoring and analyzing have shown that they outperform centralized architectures using comparable hardware. They are scalable, flexible, use standard hardware and open source software. Defects in distribution methods cause an inaccurate analysis or an unequal load balance, therefore high performance hardware is currently preferred by potential customers. Yet the fast development of the web and bandwidth causes a growing gap between network speed and computing speed. I think this gap can only be closed by introducing distributed architectures, if needed combined with high performance hardware.

Future work has to concentrate on improving dispatch methods, since this is a bottleneck. Recent approaches are not sufficient in meeting the criteria for network monitoring and analyzing. New ideas have to be tested, like this peer to peer method [10], to fully utilize given resources.

5. REFERENCES

[1] Brauckhoff, D., Tellenbach, B., Wagner, A., May, M., Lakhina, A., 2006, Impact of Packet Sampling on Anomaly Detection Metrics, 6th ACM SIGCOMM Conference on Internet Measurement, Rio de Janeiro, Brazil, October 25-17, 2006, pp 159-164.

[2] Cisco Systems inc., 2009, Cisco Visual Networking Index Forecast, White Paper

[3] DaSAHIT Project Homepage, http://www.csg.uzh.ch/ research/dasahit, March 2010

[4] Endance Homepage DAG Cards, http://www.endace.com /endace-dag-high-speed-packet-capture-cards.html, March 2010

[5] Han, S., Kim, S., Ju, H. T., Hong, J. W. K., 2002, The Architecture of NG-MON: A Passive Network Monitoring System for High-Speed IP Networks, 13th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management, Montreal, Canada, October21-23, 2002, pp 16-27.

[6] Kitatsuji, Y., Yamazaki, K., 2004, A Distributed Real-time Tool for IP-flow Measurement, International Symposium on Applications and the Internet, Tokyo, Japan, January 26-30, 2004, pp 91-98.

[7] Libpcap/tcpdump homepage, http://www.tcpdump.org/, march 2010.

[8] Mao, Y., Chen, K., Wang, D., Zheng, W, 2001, Cluster-based Online Monitoring System of Web Traffic, 3rd International Workshop on Web Information and Data Management, Atlanta, Georgia, U.S.A., November 9-10, 2001, pp. 47-53.

[9] Morariu, C., Stiller, B., 2008, DiCAP: Distributed Packet Capturing Architecture for High-Speed Network Links.

[10] Morariu, C., Stiller, B., 2007, A Distributed Architecture for IP Traffic Analysis, Lecture Notes in Computer Science.

[11] Pf_ring homepage, http://www.ntop.org/PF_RING.html. march 2010.

[12] Sallay, H., AlShalfan, K., Fredj, O. B., 2009, A scalable distributed IDS Architecture for High speed Networks, IJCSNS VOL.9 No.8, August 2009.

[13] Snort homepage, http://www.snort.org/, march 2010.

[14] Szeby, T., 2005, Statistical Sampling for Non-Intrusive Measurements in IP Networks, Ph. D. Thesis, Technische Universität Berlin, Universitätsbibliothek ,Fakultät IV Elektrotechnik und Informatik.