An Ethernet Access Architecture for Highly Available IPTV

Wei-Kuo Liao
Department of Communications Engineering
NCTU, Hsin-Chu, Taiwan

Ping-Hai Hsu Shu-Kang Tseng Kang-Chiao Ling
Information and Communications Research Laboratories
ITRI, Hsin-Chu, Taiwan

***Abstract* — We propose a practical Ethernet access architecture to support high availability and service continuity for IPTV. To reduce the per-line cost, our approach is interconnecting switches, most of which are low-cost, in a hierarchical fashion to provide broadband and cost-effective IPTV multicasting. The remaining key issue is to endow the IPTV services with high availability. To this end, we base on the redundant pair method and propose an architecture which achieves short failover process to seamlessly continue the IPTV services. In realization, our effort is restricted in extending the IGMP proxy running on the switch controller, and we describe our realization by presenting the basic design principle. To demonstrate its viability, we test our proposed architecture on ACTA which is a commercially available platform. Experiment results show that in general the failover process can be carried out within 50 ms.**

***Keywords*—Carrier grade, IPTV, first-mile Ethernet.**

1 INTRODUCTION

Ethernet in the first mile (EFM) [1] is envisioned as a key technology to carry the IP Television (IPTV) services. This vision is made realistic by observing the success of Ethernet in the local area network which features in the merits of high bandwidth provisioning but very inexpensive and easy deployment. Moreover, newer Ethernet standard [2, 3] is now proposed to meet the quality of service (QoS) requirement via the classes of services and VLAN technologies. With the cost-efficient EFM in place, the triple-play services could be made affordable for the most of the end users.

Though it is promising by using EFM to bridge the high-speed LAN and WAN, the concern is raised because Ethernet lacks the carrier-grade feature of high availability. As such, the IPTV services could be constantly interrupted or temporarily ceased due to the failure of network components. We believe that IPTV service through EFM can reach its full potential only if by doing so the IPTV service is made at least as highly available as the cable TV service.

To provide high availability, a widely applicable method is the redundant pair. That is, a redundant system is prepared in advance and performs a failover procedure to recover the ongoing services in the faulty system. In applying such technique to EFM for providing highly available IPTV (HA IPTV) services, it is desirable to recover the IPTV services within a very short period so as not to let the end users perceive the interruption. When resolving the issue, the following points are also needed to be addressed:

- To reduce the per-line/per-customer cost, interconnecting small switches with limited capability to form the whole access system is preferred. The VLAN stack and VLAN translation are assumed not supported in these switches. Only the technologies such as port isolation and fast VLAN reconfiguration are implemented. On a fault occurrence, only the IPTV service is selected to be recovered promptly.
- To further reduce the deployment cost, the high bandwidth utilization in the access system and access link is another pivot design consideration. Unlike the analog cable TV, channels of IPTV are sent separately and each consumes bandwidth ranging from 1Mbps to 13 Mbps. Typically 300 channels are estimated to be available from a provider. Besides, each broadcast television channel is an IP multicast group. To save the bandwidth, the subscriber changes the channel by leaving one group and joining a different group. In viewing that the bandwidth in the access system could be mostly consumed by IPTV services, tailoring to the broadband IP multicasting becomes a major factor in forming the switch interconnect.
- Making no assumption about the response time of upstream router outside the access system is more practical. Therefore, when a fault occurs, it is less viable to request the upstream router to redirect IPTV services over an alternative path on demand during the failover process. Relying on the spanning tree protocols to find another path for the service continuation is not viable, either. They usually take seconds or at least hundreds of milliseconds to complete the path recovery process [3]. Most importantly, they are designated to recover all the flows in the broadcast manner, and thus may cause a temporary broadcast storm.

Knowing this, we propose a practical Ethernet access architecture to achieve the low expense access system for HA IPTV. In the realization of the architecture, our effort is on coordinating the IGMP proxies in the redundant pair. To this end, we construct a software design pattern which maintains a consistent view of multicast membership information for the redundant pair and performs the necessary failover operation. The core of the software design pattern is finally identified as a core protocol to support fast health check, high-bandwidth transmission, and mutual exclusion.

The remainder of the paper is organized as follows: Sections 2, 3, and 4 describe the interconnect patterns, the software design pattern, and the core protocol, respectively.

This work was supported by National Science Council, Taiwan, under Grants NSC 96-2219-E-009.

978-1-4244-2324-8/08/$25.00 © 2008 IEEE.

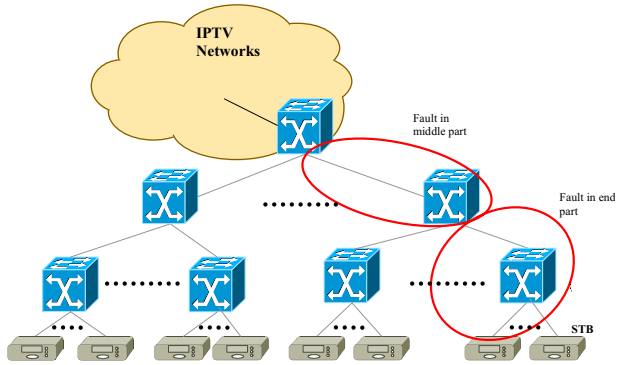

Fig. 1. A sample hierarchical switch interconnect for EFM.

We present the implementation and experiment results in Section 5. At last, we draw the conclusion in Section 6.

2 Interconnect Patterns For HA IPTV

Fig. 1 exemplifies a hierarchical switch interconnect for EFM. Each set-top box (STB) is connected to an access switch. The technique "port isolation" is applied to each switch so that the frames from a STB are forwarded upward all the way to the IPTV network and vise versa. With such a hierarchy, a clear merit is that the number of ports in a switch is limited even when the population of IPTV subscribers scales up. Moreover, by using IGMP snooping in each switch, only the channels in need by a downstream STB are necessary to be passed through the switch. Therefore, the number of channels passing through a switch in the lower level is smaller and thus lower switching capacity in such a switch is sufficient. Besides, similar to Internet Diffserv QoS architecture, the QoS requirement of downloading streams, including IPTV services, can be satisfied by only regulating them at the switch in the uppermost level and thus the other switches forwards the frame at their best effort. By reducing the switching capacity in the lower-level switches and placing the intelligence only in the switch at the uppermost level, the low-expense and high-quality IPTV service can be achieved.

Faults could occur in both links and switches. Consider the path from the upstream router in IPTV network to a STB in the hierarchical switch interconnect. We can identify two types of faults, namely "fault in middle part" and "fault in end part," as shown in Fig. 1. The fault in end part considers the fault occurring in the access link, the lowest level switch, and the link between lowest level and second lowest level switches. The fault in the middle part considers that in a switch (other than the lowest one) and its uplink. Apparently these two kinds of faults cover faults in all the components lying in the middle of the path from the upstream router to the STB.

To overcome the fault in the end part, Fig. 2 lists a dual-home scenario. Each STB now connects to two switches. To save the bandwidth of access links, an IPTV channel is delivered through only one access link to STB. A naïve approach to overcome such fault is to let STB detect the fault of the upstream hop and initiate the failover process to shift its load to the other link upon an occurrence of a fault. In addition, the switch inside the STB is so configured via VLAN that two uplinks are not in the same broadcast domain to increase the uplink bandwidth utilization. Besides, the IPTV service is

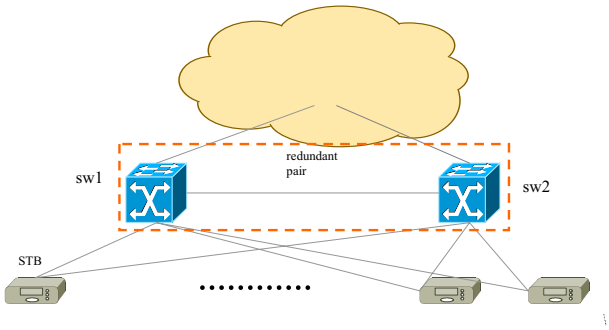

Fig. 2. Dual-home interconnect pattern to overcome fault in end part.

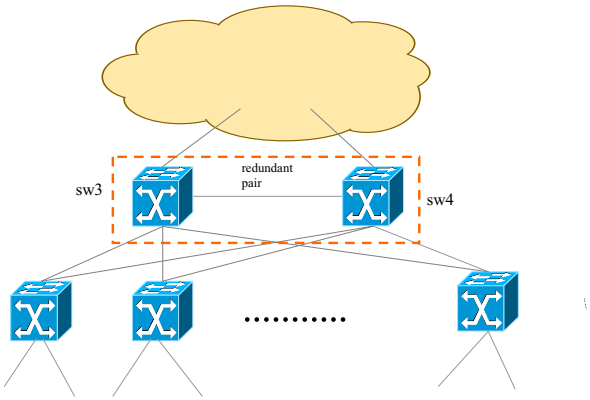

Fig. 3. Dual-hop interconnect pattern to overcome fault in middle part.

downloaded only to one of the access switches, say sw1, and thus IPTV channels are conveyed through the access link from sw1. However, after detecting fault, the failover process between switches will be not performed until the receipt of corresponding message from STB. In addition, the redirection of IPTV service to sw2 by issuing an IGMP join message to upstream switches needs extra latency. To avoid both latencies, we apply the redundant pair to both access switches. Besides, two switches receive the same channels from the upstream. We allow the IGMP messages from STB to be broadcasted to these two switches. As soon as sw2 detects a fault in sw1, sw2 simply bypasses each ongoing IPTV channel to the downstream STB in need of it. It is noteworthy that in doing so, sw2 needs to backup how the IPTV channel currently being distributed by sw1.

To provide the availability with respect to the fault in the middle part, we arrange the dual-hop pattern, as shown in Fig. 3. The switch in the lower level connects to sw3 and sw4, and the IGMP messages are broadcasted to them. Besides, the same channels are downloaded to both sw3 and sw4. The sw3 and sw4 then perform the same operations as sw1 and sw2 when a fault occurs.

To improve the robustness, we let half of channels being bypassed by one switch and the other half being the duty of the other switch. Consequently, half of IPTV service is still available if an undetected fault occurs. The concept of duty channel of a switch is then defined as the channel assigned to the switch. Doing so also accelerates the channel selection if the extra access link can be used to carry some neighboring

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE "GLOBECOM" 2008 proceedings.

channels of the one currently being watched. In this paper, we group the even channels and odd channels according to the channel index, and assign them as the duty channels for both switches in the redundant pair, respectively.

The dual-hop interconnect pattern can co-exist with the dual-home interconnect pattern. For example, suppose a switch below the redundant pair in Fig. 3 connects to the STBs. If we apply the dual-homed interconnect pattern to it, we can simply let the uplink of sw1 and sw2 ending up with sw3 and sw4, respectively. In this way, switches sw3 and sw4 treat the sw1 and sw2 as a single switch. Besides, sw3 and sw4 bypass the same those channels, which are needed by the downstream STBs, to each sw1 and sw2. Similarly, the dual-hop interconnect patterns could be repeatedly applied to hops along the path to the IPTV network.

3 Software Design Pattern For HA IPTV

We extend IGMP proxy (RFC 4605) to realize the redundant pair under the dual-hop interconnect pattern. The redundant pair under dual-home interconnect pattern can be implemented in a similar way.

Suppose that two processes are running atop the controllers of two switches, respectively. To learn how the IPTV channels being distributed by the other switch, these two processes maintain the consistent view of membership information, which is stored in the table object. Each table entry consists of fields of multicast address of the channel, status, and a list of downstream switches which need the channel at present. Besides, a dirty tag is associated with each table entry. As in the most of cache coherence protocol, the dirty tag is set until the other process confirms the update. The dirty tag is reset when the other process updates the status of the same membership to the same downstream switch. It can be easily proved that the order of update messages for the same channel to the same downstream switch is preserved in this way if two processes receive them orderly and correctly. Besides, the IPTV services function normally even when there are undetected failures in receiving IGMP messages at a single switch. If the existence of the other process has not been detected for a certain period, say 100 ms, the dirty tags will be all set to hypothesize that the fault is not transient so that the table content could be completely out of synchronization in the other process. Initially, the table is empty. Besides, the table also caches the update for membership removal to a downstream switch but will be flushed out after one second or after being confirmed by the other process.

The software design pattern in Fig. 4 depicts our basic

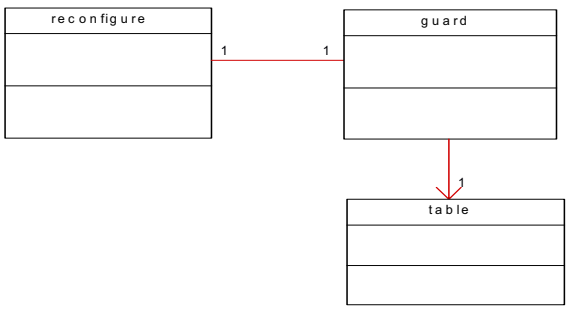

Fig. 4. The software design pattern for handling the high availability.

design principle. Basically the reconfigure object behaves like an IGMP proxy except that it needs to inquire the guard object before updating its membership database. To do so, it forwards each update of membership (e.g., IGMP report, join, or leave) to the guard object. It is the duty of guard object to update any information in the table and peer with the other process. Upon a status change of the other process or the change of table content, the guard object notifies the reconfigure object. The reconfigure will do the operation, such as modifying the forwarding database in the switch or VLAN configuration, and induce the specified channels from the upstream through IGMP messages.

With the above basic principle, our major design effort is then imposed on the guard object. The instructions given by the guard object to the reconfigure object is “delete a member in a port”, “insert a member in a port”, “unblock the non-duty channels”, “unblock the duty channels”, “block non-duty channels”, and “block all channels.” The IGMP join or leave will be also sent by the reconfigure to the upstream accordingly, e.g., if it is a new member found in the IGMP proxy, an IGMP join message will be issued. There are four states associated with the guard object eventually, i.e., *startup*, *normal*, *resuming*, and *alone*. We describe the behaviors of states in the followings: In startup state, the delivery of all channels is blocked. When the existence of the other process is detected within one second, then the guard object enters into the resuming state. Otherwise it transits to the alone state.

In resuming state, if there is any dirty table entry, then the guard object updates the table content in the other process. The guard object stays in the resuming state for one second to ensure that all the needed channels are already transferred from the upstream. After that, it enters into the normal state. On transiting into the normal state, the guard object request the reconfigure to block the non-duty channels and distribute the duty channels. In normal state, the table content is also needed to be synchronized.

When a failure for the other process is reported, the guard object transits into the alone state. When guard object stays in the alone state, it instructs the reconfigure to do the operation as a stand-alone switch, i.e, the duty and non-duty channels are allowed to pass to downstream. After the other process is detected again, the guard object enters into the resuming state.

4 Protocol for Guard Object Peering

When examining the “guard” object more closely, we find that it requires the functions of fast health check, mutual exclusion for table consistency, and high-throughput/orderly-transmission. Because these operations depend on each other, e.g., the delivery and mutual exclusion relies on the result of health check, it is nature to integrate all the functions into one protocol. Based on such a protocol to form the peer relation between guard objects in two respective processes, the remaining task of the “guard” object includes straightforward jobs only, i.e., to handshake with the reconfigure, scan the table, and update the table in local or remote process.

It is noteworthy that the conventional communication protocol for fault tolerance is designed for more than two components participating in the protected group [5]. As a

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE "GLOBECOM" 2008 proceedings.

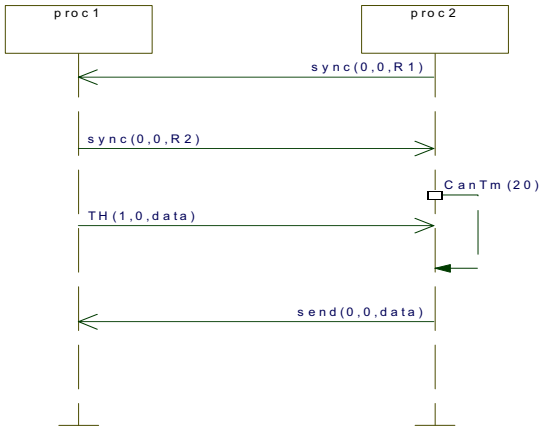

Fig. 5. Connection setup scenario.

consequence, they emphasize on the reliable broadcast, atomic broadcast, and casual broadcast to ensure the ordering of the messages delivered and received. In our targeted environment, our communication is only one-to-one. Some issues are thus resolved easily. Besides, some other features can be designed more efficiently, such as concurrency control. Most important of all, our protocol should report the condition of the other process in a very short period, which is not addressed in the previous proposals.

The protocol basically combines the sliding window, fast hello, and a simplified token passing algorithm [6]. The sliding window protocol is to orderly deliver the data between the redundant pair in a high-throughput fashion. The fast hello is dedicated to fast health check. The token passing algorithm is to enforce the mutual exclusion, i.e., whichever wants to enter into the critical section needs to grab the token in advance. As a consequence, our protocol needs to create the token, pass the token, and report failure condition if the token status is weird, i.e., lost token or duplicated token.

The typical communication environment we are encountered is two switches connected via a short-delay and highly reliable network where the packet error rate is very small. The exchanged message format is listed as follows:

- n-bit sequence number;
- n-bit expected sequence number;
- 1-bit sync field;
- 2-bit token status;
- payload.

The sequence number field is for realizing the sliding window ARQ. To simplify our discussion, we demonstrate our protocol by using 1-bit sequence number, i.e., the ARQ is stop-and-wait. The expected sequence number field identifies the sequence number of the message that the sender expects to receive next. If the sync field is set, the sender indicates its intent to set the sequence number to zero in both sides. The two bits of token status tells us whether the sender is holding a token (TH), requesting a token (TR), or otherwise (send).

In the followings, we list three major scenarios for the protocol, namely connection setup scenario, timeout scenario, and token loss scenario. The scenario for the duplicated token is similar to the token loss scenario

The first scenario, as shown in Fig. 5, considers the connection setup between two processes. Here we use the term

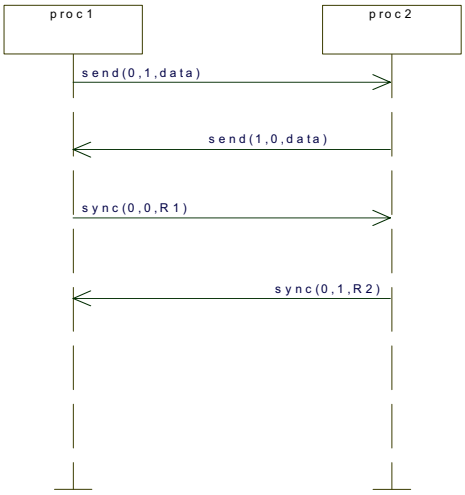

Fig. 6. Token-loss scenario.

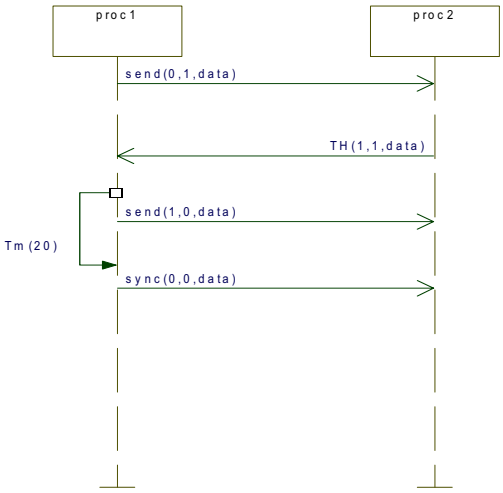

Fig. 7. Timeout scenario.

"process" to indicate that it is the application rather than the switch to be protected. The connection setup procedure consists of synchronization where each process needs to generate numbers and the larger one grabs the token first. The process with larger number then sends the message with TH indicated until the token is passed to the other process. If the process with smaller number does not need the token for the time being, it simply sends the message whereas piggybacking the data is allowed. On the other hand, if it needs the token, it sends the message with TR set. Upon the receipt of a send message without the TH set, the process then grabs the token.

After connection setup, the situation where the token possessed by one process and messages arrive in time is regarded as the normal condition. Otherwise, a failure will be reported. As shown in Fig. 6, the process which sends the message without TH or TR set receives the message where the TH is neither set. The situation claims that the token is lost and a failure must be reported and then the connection setup scenario is entered immediately. Fig. 7 shows the timeout scenario where the message is not received in time. In such a case, a fault is also reported and then the connection setup scenario is re-entered promptly.

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE "GLOBECOM" 2008 proceedings.

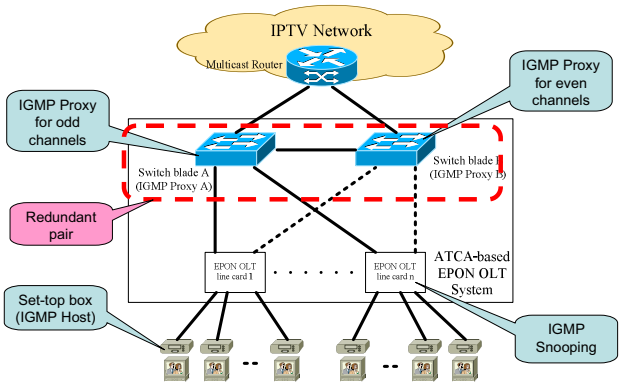

Fig. 8. ATCA-based EPON OLT system architecture for IPTV service.

	Failover time (sec)
1	0.045769
2	0.044277
3	0.045042
4	0.045483
5	0.048496
6	0.043800
7	0.047309
8	0.044403
9	0.046180
10	0.045776
11	0.043100
12	0.043385
13	0.044033
14	0.043504
15	0.044129
16	0.042722
17	0.045161
18	0.044966
19	0.044003
20	0.043727
Average	0.044763

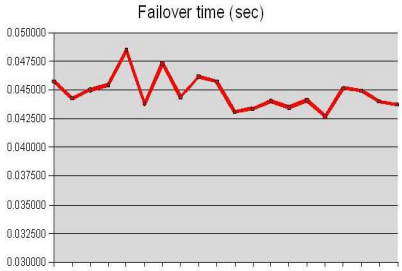

Fig. 9. The results of failover time.

It is noteworthy that the starvation of token is possible since the process who grabs the token may ignore the request by the other process. Therefore, the "guard" object reports a failure if the token request is not responded for a certain period. The deadlock of waiting for token is not possible because in such a case a failure is reported due to token loss and both processes start all the way from the connection setup.

To enable fast health check during the period when there is no data to be transferred, the sender is allowed to send the null-data message without sync being set. The receiver simply ignores the sequence number field and regards the message as the one for reporting the health condition only.

In our implementation, the message is encapsulated in UDP and the sender sends out at least one message within ten milliseconds.

5 Implementation and Experiment Results

To show that our architecture is viable, we implement the dual-hop interconnect pattern in Advanced Telecommunications Computing Architecture, known as ATCA, which is a new system form factor defined by the PCI Industrial Computers Manufacturers Group (PICMG). The ATCA provides an industry standard platform that enables building telecommunication grade products in a multi-vendor compatible environment. ATCA is the first standardized platform for high availability system with redundant power, cooling, and high-speed interconnections for data and control plane.

Fig. 8 shows the system architecture to test our implementation. We generate two identical testing flows with the same content, speed (1.5Mbps), frame length, and with simultaneous start and stop operations by smartbit. These two flows are treated as "odd-channel" and "even-channel" respectively and served by Switch blade A and B respectively in normal case. In our VLAN configuration, two different VLAN tags are dedicated for the odd and even channels, respectively. In the case without any failures, Switch blade A and B can serve the same amount of traffic within a period. If a failure occurs on Switch blade B, the "even-channel" service will be out of service for a short period, then the service is resumed after Switch blade A taking it over. However, the "odd-channel" is still running without any interruptions at that moment. It is clear that the difference of the amount of packet between "odd-channel" and "even-channel" can be easily transformed into failover time via dividing the difference of bytes received by the data rate of the individual flow.

We incorporate the software design pattern into the IGMP proxy running atop Linux operating system on the switch controller. The results of twenty experiments are shown in Fig. 9. The average failover time is about 45ms. As shown, the performance is always better than 50ms.

6 Conclusion

We consider how to equip EFM to support HA IPTV by the technique redundant pair. The key challenge is that we have to cut down our deployment cost while keeping the short failover time. By observing that an IPTV broadcast channel is indeed a multicast group, we propose a practical Ethernet access architecture which is basically dedicated to support highly-available IP multicasting. We implement our idea in ACTA and the results show that our proposed architecture is a viable approach.

In this paper, we confine the operations of failover process in the redundant pair. Our future work is to explore the potential of more cooperative activities between switches in access system to reduce cost whereas improving the availability of IPTV services.

References

[1] Ashwin Gumaste, Tony Antony, First Mile Access Networks and Enabling Technologies, Cisco press, Feb. 2004.

[2] Girish Chiruvolu, et al., "Issues and Approaches on Extending Ethernet Beyond LANs," IEEE Communication Magazine, pp. 80-86, March 2004.

[3] S. Khandekar et al., "Metro Ethernet Network QoS Framework," work in progress, MEF, Nov. 2003.

[4] Elie Sfeir, Saiidrine Pasquahi, Thomas Schwabe, Andreas Iselt, "Performance Evaluation of Ethernet Resilience Mechanisms", IEEE, 2005

[5] Pankaj Jalote, Fault Tolerance in Distributed Systems, Prentice Hall, 1994.

[6] M. Ben-Ari, Principles of Concurrent and Distributed Programming, 2nd ed., Addison-Wesley, 2006.

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE "GLOBECOM" 2008 proceedings.