PeerIIR: Peer-to-Peer Interactive Internet Radio System

Tzu-Chieh Tsai, Tong-Yen Hsieh, Wen-Ching Lo
Department of Computer Science
National Chengchi University,
Taiwan, Taipei,
ttsai@cs.nccu.edu.tw, g9603@cs.nccu.edu.tw, d9806@cs.nccu.edu.tw

Abstract—**Peer-to-Peer (P2P) applications are popular recently and have become one of the hottest research topics. The participants can share their resources (such as processing power, disk storage, and network bandwidth) in the P2P architecture to collaborate file downloading and streaming services. In this paper, we design and implement an Interactive Internet Radio system using the P2P approach, called PeerIIR. When the host, co-hosts, and calliners are speaking at the same time, they will produce multiple streams which need to deliver to all the audience on the system. This will consume the network bandwidth inefficiently, or even exhaust the link capacity of the audience. Thus, how to process multiple streams produced at the same time and to deliver to all the audience efficiently is the key issue. When there is only one program host producing the audio stream, a distribution tree is built to distribute it. If there are co-hosts or calliners speaking, a distributed mixer negotiation algorithm is performed to build a voice mixing tree among PeerIIR servers. Therefore the audio streams are mixed distributedly and step by step along the mixing tree to save transmission bandwidth. The results from series of simulation show that the performance for response time and link/node stress is enhanced compared with some related works.**

Keywords-Peer-to-Peer; Internet Radio; Voice-over-Internet-Protocol (VoIP); Call-in Service.

I. INTRODUCTION

Using Peer-to-Peer (P2P) overlay has become an increasingly popular approach due to its scalability and easy deployment. In the P2P architecture, each peer can share their resources without the need of a central server to collaborate file downloading or streaming services. Several P2P-based streaming systems have been successfully deployed to date, such as PPlive[1], PPStream[2], and Skype[3]. We can watch TV shows via PPstream or PPlive. We can make a phone call via Skype. We can also hear an audio program on the Internet like traditional radio programs. We call such audio service transmitted via Internet as "Internet Radio".

Several Internet radio systems have been developed, such as live356[4], PTTRadio[5], and ipavo[6]. These systems broadcast the stream from the host to all audience. Specifically, these systems can only process one stream at a time. However, many live interactive shows which allow audience to call in to participate in the program such as political commentary shows, may produce multiple streams concurrently. In this case, hosts and audience can interact like a conference call. Furthermore, the P2P approach has to be utilized to reduce the possible tremendous broadcast streaming. Thus, in this paper, we will design a Peer-to-Peer Interactive Internet Radio system, called PeerIIR. In our PeerIIR system, everyone could be a program host to perform a show. The host can invite co-hosts to make the show together, and can allow audience to use the call-in service. When there is only one program host producing the audio stream, a distribution tree is built to distribute the audio stream on a P2P basis. If there are co-hosts or calliners speaking, a voice mixing tree is built among the designated PeerIIR servers. Therefore the audio sources are first mixed locally and then further mixed together among them to save transmission bandwidth.

The rest of this paper is organized as follows. Section 2 introduces related works in P2P streaming systems and multi-party VoIP conferencing systems. Our PeerIIR system and algorithm are proposed in Section 3. In Section 4, we present the simulation results compared with some related works. Section 5 concludes this paper and remarks on the future work.

II. RELATED WORKS

There are several existing approaches to deal with multiple streams produced at the same time. [7] shows the overlay multicast approach to distribute multiple audio streams concurrently. Although the overlay multicast is well suited for broadcast applications with one speaker, it becomes inefficient for the application with multiple streams produced at the same time. The system may be overloaded by processing many audio streams simultaneously. In addition, the system has to maintain large number of multicast trees for all speakers. Thus, it will incur a lot of maintenance overhead.

The audio mixing scheme can effectively reduce the number of concurrent streams and conserve the bandwidth. However, centralized audio mixing lacks the scalability and the mixer could be a bottleneck. Previous work has proposed distributed mixer processing (DMP) approaches (e.g., [8], [9], [10]) that distributedly mix the audio streams step by step along the mixing tree. This system not only saves the network bandwidth, but also balances the loading of mixers.

III. PEERIIR: PEER-TO-PEER INTERACTIVELY INTERNET RADIO SYSTEM

A. System Model

In our PeerIIR system, we build distribution trees and a voice mixing tree for an audio program channel. When there is only one program host producing the audio stream, a distribution tree is built to distribute it. If there are co-hosts or calliners speaking, a distributed mixer negotiation algorithm is performed to build a voice mixing tree among PeerIIR servers. Figure 1 depicts the PeerIIR system model.

CHINACOM 2010, August 25-27, Beijing, China

DOI 10.4108/chinacom.2010.61

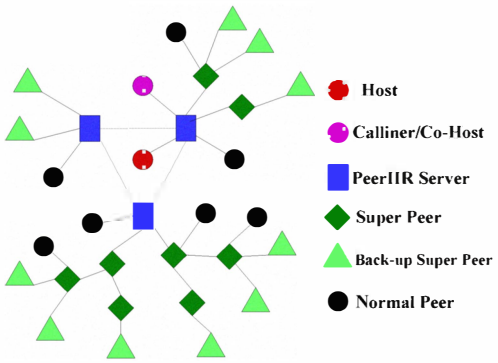

Figure 1: PeerIIR System Architecture.

There are six major components in PeerIIR system:

1. (Program) Host: the main content provider of a program. The host can allow peers to use the call-in service or invite co-hosts to participate in the program.
2. Calliner/co-host: a peer which uses the call-in service/the assistant content provider of a program.
3. PeerIIR server: a dedicate server which provides the advertisement and the service message. It can also deal with the program host leaving the system without pre-notice. The PeerIIR server also maintains super peer list.
4. Super peer: any normal peer with a public IP address having sufficient network bandwidth selected to become a super peer. It is a distributor which distributes audio streams to multiple peers through the tree.
5. Back-up super peer: the children peer selected by a super peer. If a super peer leaved, the back-up super peer will replace it and send the audio stream to its children.
6. Normal peer: the general audience.

PeerIIR assigns at least one PeerIIR server to each program channel. PeerIIR servers are the root of distribution trees. When the speaker is only the program host, it sends the audio stream to all the PeerIIR servers which then transmit to audience via distribution trees. We assign a PeerIIR server to the program host by "host ranking". *Host ranking* is derived from (1) the popularity of the host's previous programs, (2) the comments from the audience. The higher host ranking means the higher attraction of this host. In this case, we assign more PeerIIR servers to it.

Someone may join the channel before the show started. At this time we can send the advertisement via the PeerIIR server until the show starts. During the show, when more peers join the channel, the PeerIIR could choose more super peers to balance the load or enhance the performance of the streaming delivery. The activity of super peers is maintained by the PeerIIR server. A new joining normal peer will request to a PeerIIR server and download the super peer list. After that, it will select a super peer to join the distribution tree and thus listen to the channel.

B. Super Peer Selection

Any normal peer with a public IP address having sufficient network bandwidth is a candidate to become a super peer. Every super peer will, if possible, choose two back-up super peers from its children. Once a new joining peer connects to the super peer, this super peer will compute the super peer score (SP) by the following equation (1).

$$SP = a * BW + b * F + c * LT, \quad (1)$$

where the BW is bandwidth, F is frequency, LT is listening time, and a, b, and c are the parameters. Different from most P2P systems, we choose the frequency and listening time to select super peers. The frequency is the feverish of this peer to the host. The higher frequency and listening time may decrease the super peer churn rate. Thus super peer chooses two back-up super peers from its children according to the super peer scores.

The overall load of a super peer may be saturated. This can be detected by the Saturated-Notification scheme. These two back-up supers will be transformed to the super peer and increase the distribution tree height. Thus, the ability of back-up super is not only back-up the super peer, but also prepares to become a super peer. We will describe the Saturated-Notification scheme later.

C. Peer Join

Figure 2 shows the peer join procedure. There are three steps: (1) After login to the login server and obtaining the PeerIIR server list, the new peer P will first send a REQUEST message to the PeerIIR servers. On receiving the message, PeerIIR servers send the REPLY message to P. (2) P selects a PeerIIR server with the minimum round trip time (RTT) as its entry point to the system and then downloads the super peer list from the selected PeerIIR. The RTT is the time measured from sending REQUEST to receiving REPLY. (3) P sends the REQUEST message to the random subset of the super peers as its parent. On receiving the message, super peers will return messages with D(SP). The D(SP) is delay time between the host to super peer. Every super peer measures D(SP) periodically. Finally, P chooses the super peer with minimum (RTT+D(SP)) to join the distribution tree.

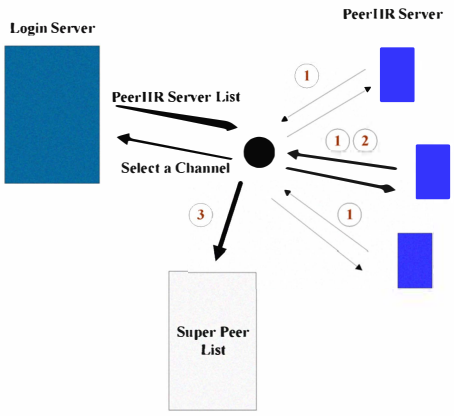

Figure 2: Peer Join Procedure.

D. *Peer Leave*

i. Normal peer: when a normal peer leaves the system, the super peer sets its status to be inactive. If this peer does not rejoin the system for a long time, the super peer will delete the information of this peer.

ii. Back-up super peer: if a back-up super peer leaves the system, its super peer will choose a new back-up super peer according to the super peer score by equation (1).

iii. Super peer: every super peer selects two back-up super peers and sends the address of back-up super peer to its parent and children. When super peer leaves the system, the back-up super peer will replace the super peer to continuously transmit the audio stream.

E. *Super Peer Maintenance*

During the running of the system, peers join and leave the system frequently. This high churn rate may cause that some super peers manage very large number of peers, but the others manage few peers. To make the loading of super peers more balanced, we design a scheme called Saturated-Notification scheme to avoid super peer overloaded. On the other hand, we also merge the light loading super peers to decrease the height of the distribution trees.

Saturated-Notification Scheme: we first define two thresholds: (1) lock threshold, which means the upload bandwidth of the super peer is nearly saturated, (2) un-lock threshold, which means the locked super peer has sufficient upload bandwidth now. While the capacity of super peer is over the lock threshold, it will send the Saturated-Notification to its PeerIIR server. This notification includes the address and peer IDs of its back-up super peers. Once receiving the Saturated-Notification, the PeerIIR server will update the super peer list: add new super peers to the list, and set the flag of the saturated super peer to be locked. Figure 3 shows the Saturated-Notification scheme.

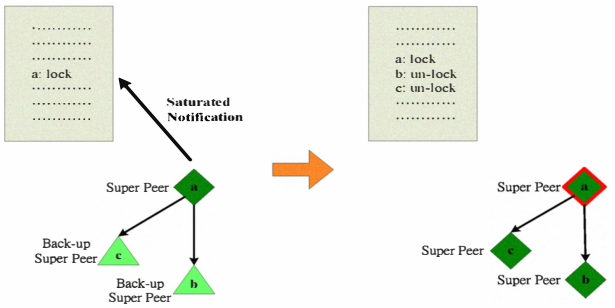

Figure 3: Saturated-Notification Scheme.

When the upload bandwidth of locked super peer is below the un-lock threshold for a while, it will request the PeerIIR server to un-lock it.

Super Peer Merge: when super peers are light loaded, we merge the super peers to decrease the distribution tree height. We define a threshold called merge threshold, which means the super peer has sufficient free bandwidth. When the upload bandwidth of a super peer is below the merge threshold, it will send the merge message to its parent super peer or child super peers. If one of them is also below the merge threshold, we merge these two super peers. Figure 4 shows the super peer merge example.

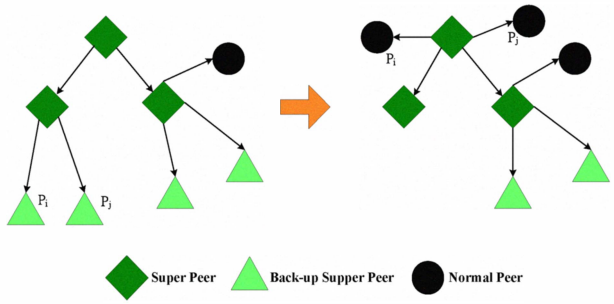

Figure 4: Super Peer Merge.

F. *Call-in Service*

Audience could use the call-in service to interact with the host. In order to use call-in, audience must run the call-in signaling first. Figure 5 shows the call-in signaling. The calliner sends the call-in request and its information to the host. If the call-in request is accepted, the host will send the information of calliner to the PeerIIR server. Then, calliner will connect to the PeerIIR server directly in order to increase synchronization and interactivity among hosts and calliners.

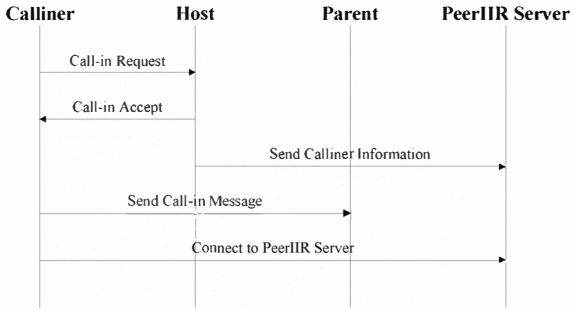

Figure 5: Call-in Signaling.

The calliners and hosts may speak at the same time and produce multiple streams concurrently. We adopted the Distributed Mixer Processing (DMP) to build a mixing tree among PeerIIR servers. This scheme will distributedly mix multiple streams and then transmit the mixed stream via the distribution trees. We consider the workload of PeerIIR servers to build the mixing tree. PeerIIR servers measure the workload periodically which is calculated by the following equation (2).

$$\text{Workload} = 0.5*\text{IS} + 0.5*\text{MTL} \quad (2)$$

The IS is the number of the mixed input streams and the MTL is mixing tree level. The higher IS means the higher mix loading of the PeerIIR servers. The higher MTL means the higher overlay hop count. So we choose these two criteria to build the voice mixing tree.

When a PeerIIR server has a new calliner connection, it will use Distributed Mixer Negotiation to join the voice mixing tree. First, the new joining PeerIIR server will send the

JOIN message to other PeerIIR servers. On receiving the JOIN message, these PeerIIR servers will return the REPLY message with the current workload. Then the new joining PeerIIR server connects to the PeerIIR server which has the minimum workload. Figure 6 shows the Distributed Mixer Negotiation.

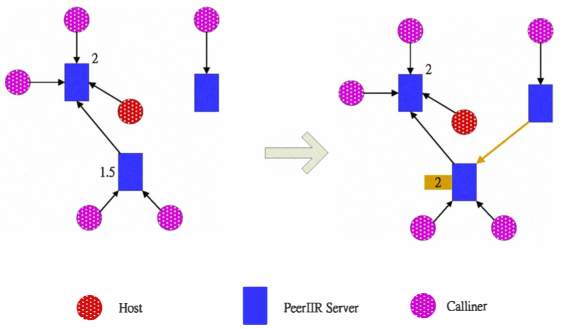

Figure 6: Distributed Mixer Negotiation.

IV. SIMULATIOIN RESULTS

A. *Simulation Setup*

We compare our PeerIIR system with the other existing approaches: (A1) Overlay Multicast, (A2) Centralized Mixing, and (A3) Distributed Mixer Processing (DMP). The simulation model is implemented in JAVA. The network setups are shown in Table 1. We take into account the asymmetric property of the residential access network where the upload bandwidth is smaller than the download bandwidth.

Table 1: Simulation Parameters.

Parameters	**Default Values**
Physical Link Delay	8~12 (ms)
Hops	1~7
Upload bandwidth	64K, 256K, 640K (bps)
Download bandwidth	256K, 1M, 2M, 8M (bps)
Mixing Delay	In * [1, 20] + [3, 15]

We use the following metrics to evaluate the quality of our PeerIIR system: (1) Average tree length of a channel which is defined as the mean tree length from all speakers to all audience; (2) Response time of a channel which is defined as the mean response time from all speakers to all audience, including the mixing delay, distribution delay, and queueing delay; (3) link stress over all physical links which is defined as required bandwidth divided by the available bandwidth. The higher link stress implies large queueing delay and loss rate; (4) node stress over all peers which is define as total amount of audio data the peer needs to process over its processing capacity. The larger node stress implies larger stream processing delay and loss probability.

B. *Simulation Results*

1) *Scenario 1*

In the first scenario, we evaluate the performance of the PeerIIR system under different number of audience in a channel, illustrated by Figure 7 ~ Figure 10. The number of audience is ranged from [100, 700]. There are three speakers for this scenario. Figure 7 shows the average tree length achieved by different approaches. We observe that the centralized mixing and the overlay multicast both have better performance than PeerIIR and DMP. This is because we don't take into account the mixing delay and the affect of the limited bandwidth. Figure 8 shows the average response time achieved by different approaches. The results show that PeerIIR achieves lower response time than the other approaches.

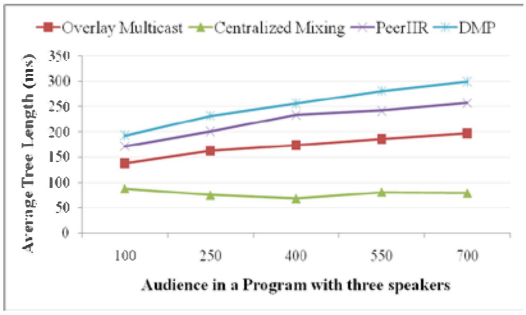

Figure 7: Average Tree Length under Different Number of Speakers.

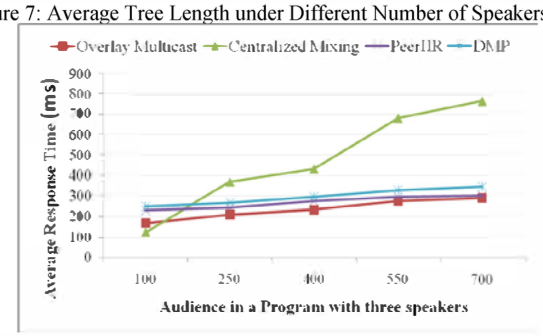

Figure 8: Average Response Time under Different Number of Audience.

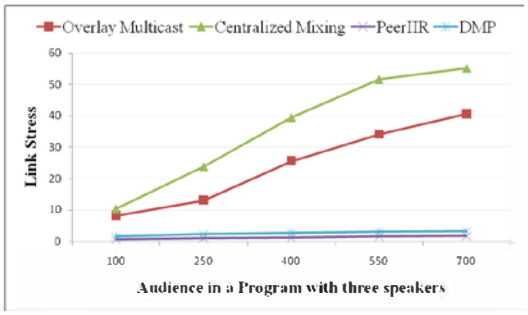

Figure 9: Link Stress under Different Number of Audience

Figure 9 shows the average link stress on all the physical links under different approaches. We observe that peerIIR can achieve similar link stress as DMP by employing explicit load balancing. The reason is that they both employ a multiple stream mixing phase that can greatly reduce the number of concurrent audio streams distributed across networks. Figure 10 shows average node stress on all the peers under different approaches. We observe that peerIIR can achieve smaller node stress due to its load balance scheme.

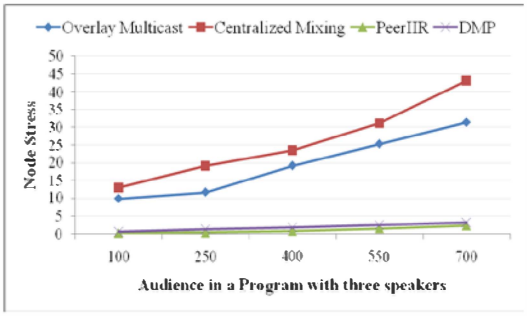

Figure 10: Node Stress under Different Number of Audience

2) Scenario 2

Different with scenario 1, we evaluate the performance of the peerIIR system under different number of speakers, illustrated by Figure 11 ~ Figure 14. The number of speakers is ranged from [5, 25] and we set the audience as 500 peers.

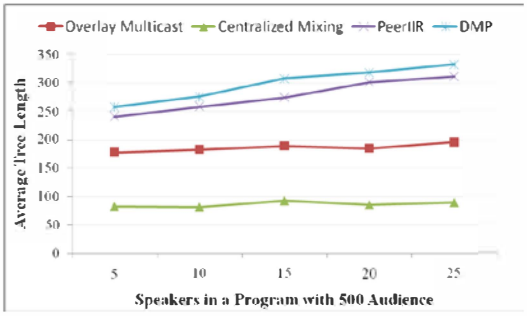

Figure 11: Average Tree Length under Different Number of Speakers.

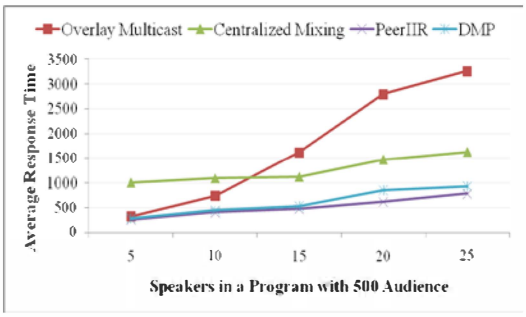

Figure 12: Average Response Time under Different Number of Speakers.

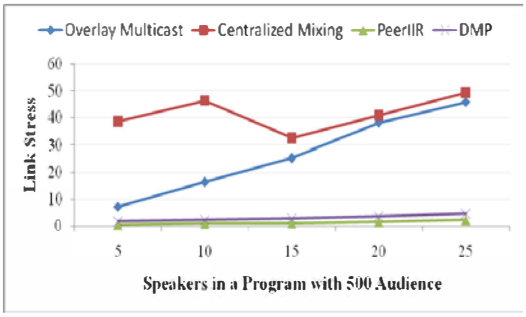

Figure 13: Link Stress under Different Number of Speakers.

Figure 11 is similar to Figure 7 because we don't take into account the mixing delay and the effect of the limited bandwidth. Figure 12 shows the average response time achieved by different approaches. We observe that PeerIIR can consistently achieve lower response time than the other approaches. Figure 13 shows that both PeerIIR and DMP have much lower link stress than the others approaches by employing distributed audio mixing. Figure 14 shows that PeerIIR can achieve lower node stress than the other approaches because of its inherent load balancing capability.

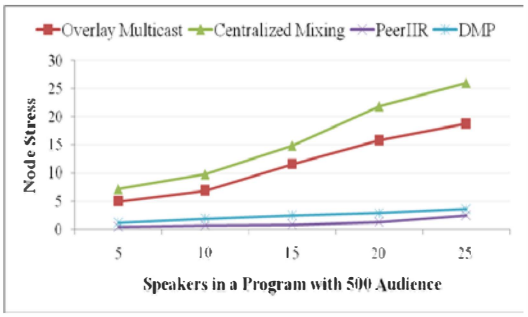

Figure 14: Node Stress under Different Number of Speakers.

V. CONCLUSION

In this paper, we design a Peer-to-Peer Interactive Internet Radio system called PeerIIR. In PeerIIR, everyone could be a program host to make any programs. The host can invite co-hosts to make the program together. The host can also allow audience to use the call-in service. By using the call-in service, audience can interact with the host. The host, co-hosts and calliners may speak at the same time and produce multiple streams concurrently. Multiple streams may burden our system due to bandwidth limitation. So we use the Distributed Mixer Negotiation scheme to distributedly mix the streams. This scheme not only conserving the bandwidth but also avoid overload of the mixers. We compare our PeerIIR with the overlay multicast, centralized mixing, and distributed mixer processing (DMP) schemes. We observe that PeerIIR can achieve the best performance than the other approaches. In the future, we will test it in the real world and target at 1000 people playing our system concurrently. We have implemented the prototype system, and the video clips of the demo can be found: http://www.youtube.com /user/NCCUmclab1

REFERENCES

[1] http://www.pplive.com/
[2] http://www.ppstream.com/
[3] http://www.skype.com/
[4] http://www.live356.com/
[5] http://pttradio.net/
[6] http://www.ipavo.com/
[7] Y. H. Chu, S. G. Rao, S. Seshan, and H. Zhang, "Enabling Conferencing Applications on the Internet using an Overlay Multicast Architecture," in SIGCOMM'01, San Diego, California, USA, August. 2001.
[8] M. Radenkovic, and C. Greenhalgh, "Multi-party Distributed Audio Service with TCP Fairness," in Multimedia'02, Juan-les-Pins, France, December. 2002.
[9] T. K. Chua, and D. C. Pheanis, "Bandwidth-Conserving Real-Time VoIP Teleconference System," in ITNG'06, 2006.
[10] X. Gu, Z. Wen, P. S. Yu, and Z. Y. Shae, "peerTalk: A Peer-to-Peer Multi-Party Voice-Over-IP System," in Parallel and Distributed Systems, IEEE Transaction on April. 2008.