Its own transmission order. The RTXLIST was implemented using the status variable in the SKB data structure. When the status variable was set as LOST, the packet was considered to be in the RTXLIST. Sequential traversing of the list allows for determination of those packets in the RTXLIST. The \textit{dupCnt} variable was also declared in the SKB in order to implement the per packet acking process. Whenever a duplicated ACK arrived, the \textit{dupCnt} of the corresponding SKB increased by one up to the threshold of 3. Every SKB having the \textit{dupCnts} value was marked as LOST.

Our implementation is advantageous as follows. First, by minimizing the restructuring of the SKB data structure, most of existing code can be reused without major code development. Most of existing TCP codes were reused except the parts related to packet recovery and retransmission. Second, byte sequential ordering of packets is needed when a packet is retransmitted due to RTO timeout. By maintaining the sequential ordering, the RTO timeout operation can be easily conducted.

\begin{figure}[htp]
\centering
\begin{subfigure}[b]{0.4\textwidth}
\includegraphics[width=\textwidth]{original_scoreboard}
\caption{Original Scoreboard}
\end{subfigure}
\begin{subfigure}[b]{0.4\textwidth}
\includegraphics[width=\textwidth]{modified_scoreboard}
\caption{Modified Scoreboard}
\end{subfigure}
\caption{Implementation of Two Lists}
\end{figure}

4.2 Fine-Grain RTT and RTO Timer Setting

In order to implement \textit{per packet RTT measurement}, which is used for \textit{per RTT congestion control} and for \textit{fine-grained RTO timer setting}, we need to identify the packet generating the current incoming ACK.

Initially, we only considered the Linux TCP timestamp for the above purpose. However, we found this is not sufficient. Some packets belonging to a single window may sometimes have the same timestamp value due to the granularity of the Linux TCP timestamp, on the order of tens of milliseconds.

Additional information should be used to differentiate the packets with the same timestamp. The variations in cumulative ACK number and SACK blocks may be a solution. For example, the increase in the cumulative ACK number correctly determines which packet generates this ACK. In the case of a duplicated ACK, the change in SACK blocks also provides information on the increase in the cumulative ACK. Therefore, we need to remember a few of the consecutive incoming ACKs to trace the change in acking. Combining the differences with the timestamp, the packet corresponding to an incoming ACK can be determined.

For rapid implementation, however, we simply included \textit{transmission order} in the TCP header instead of maintaining a few of ACKs. If the packet arrives at a TCP receiver, the receiver echoes the transmission order value in its replying