```java
public boolean routeEvent(EventHolder event, SocketAddress fromAdr)
{
  RoutingEntry entry;
  ModeHolder modeHolder;
  int variableId = event.m_event[0].getInt(Constants.EVENT_VARIABLE_ID_OFFSET);
  long timeStamp = event.m_event[0].getLong(Constants.EVENT_CREATED_OFFSET);

  // Do we route this variable ?
  if ((entry = (RoutingEntry)this.m_tbl.get(variableId)) == null)
    return false;

  // Do we route this variable in the current operating mode ?
  if ((modeHolder = entry.m_modeTbl[this.m_currentMode]) == null)
    return false;

  // Do we flood the event ?
  if (modeHolder.m_flooding > 0)
  {
    ........
    return true;
  }

  IntervalHolder intervalHolder;
  RoutingHolderSparse routingHolderSparse;
  OutHolder outHolder;
  int missed = 0;
  int sentTo = 0;

  for (int i = 0; i < modeHolder.m_sparseHolders.size(); ++i)
  {
    intervalHolder = (IntervalHolder)modeHolder.m_sparseHolders.get(i);

    // Should this interval be forwarded
    if ((created + entry.m_pubIntervalHalf) %
         intervalHolder.m_currentIntervalPath.m_smallestSubInterval < entry.m_pubInterval)
    {
      for (int j = 0; j < intervalHolder.m_sparseHolders.size(); ++j)
      {
        routingHolderSparse = (RoutingHolderSparse)intervalHolder.m_sparseHolders.get(j);

        // Have we already marked this one and should it be marked ?
        if (routingHolderSparse.m_outHolder.m_send == false &&
          ((created + entry.m_pubIntervalHalf) %
           routingHolderSparse.m_smallestSubInterval < entry.m_pubInterval))
        {
          routingHolderSparse.m_outHolder.m_send = true;
          ++sentTo;
        }
      }
    }
  }

  event.incrementRef(sentTo);

  // Send the event
  for (int i = 0; i < modeHolder.m_sparseOutInterfaceHolders.size(); ++i)
  {
    outHolder = (OutHolder)modeHolder.m_sparseOutInterfaceHolders.get(i);

    if (outHolder.m_send)
    {
      // Is this an alert
      if (routingHolder.m_priority == CommConstants.PRIORITY_ALERT)
      {
        if (!outHolder.m_outInterface.pushAlertEvent(event))
          ++missed;
      }
      else if (!outHolder.m_outInterface.pushEvent(event, outHolder.m_priority))
        ++missed;

      outHolder.m_send = false;
    }
  }

  // Make sure that we don't encounter mem. leaks
  if (missed > 0)
  {
    if (event.decrementRef(missed) == 0)
      return false; // This event must be recycled
  }

  return true; // Event is sent|filtered and memory management is completed
}
```