PROACTIVE ADHOC NODES FOR REAL-TIME VIDEO

Tien Pham Van

Heinz Nixdorf Institute, University of Paderborn, Germany, Email: vantien@uni-paderborn.de

ABSTRACT

Transmission of real-time video over wireless adhoc networks copes with numerous issues due to its bandwidth-greediness and time-sensitiveness, as well as the nature of the networks. This paper introduces a proactive architecture where intermediate nodes do not only simply forward video packets, but are also responsive to the communication. We design a special "cache" at each node to shorten retransmission paths. Each proactive node can analyse semantics and timing information of retained packets to make the most appropriate decisions on forwarding, aiming at minimizing playback distortion and power consumption. Both theoretical analysis and experimental results collected from a real-life testbed demonstrate the feasibility and the outperformance of our proposed framework.

1. INTRODUCTION

Resource limitation, node mobility, and environmental instability in adhoc networks are challenging against real-time video. Every loss of packet causes extra distortion, not only to the associated frame but also to the dependent ones. For instance, in the MPEG4 standard, an I frame is mandatory to decode all frames within the same GoP (Group of Picture). Any P frame cannot be decoded without the previous P (or I) frame, while a B frame depends on both its adjacent P (or I) frames. To recover erroneous packets, FEC (Forward Error Correction) may be deployed. In this technique, considerable redundant data and computation load are needed for detecting and correcting errors. Consequently, introduction of pure FEC is likely unbearable; FEC should instead be used in combination with other error-relief strategies [1][2][3]. In [2], MDMC (Multiple Description Motion Compensation) is proposed to deal with packet loss by transmitting multiple independent substreams. This strategy however reduces the coding efficiency and is tightly codec-dependent. Both FEC and MDMC only work efficiently when packet loss is mild [1][2].

As supported by adhoc routing protocols, video traffic can be distributed over multiple paths concurrently, which hence increases the aggregate throughput [2][4]. In wireless adhoc communication, severe packet loss may occur frequently. Loss of important frames makes transmissions of their dependent ones meaningless. Consequently, retransmission for lost packets (especially, of important frames) is highly desired to reduce the loss rate and to efficiently exploit the available bandwidth that the path diversity offers [2][4].

While retransmission effectively improves the perception quality, it obviously causes further latency [2][4]. All ARQ requests have to reach the very sender to be processed. Generated from the sender, retransmitted packets must traverse all the hops down to the receiver. In other words, the sender and the receiver must drive all the process of communication while each intermediate node simply relays packets to the next hop. This passive mechanism does not only add extra delay, but also reduce the effective bandwidth of the whole path. Furthermore, it does impose additional workload on the sender, which is busy with encoding video.

Dealing with the issue, we design a novel architecture in which intermediate nodes can understand the semantics of video packets and hence know how to select packets to forward, taking minimization of distortion and the overall power consumption into account. By allocating a small "cache" for retaining video packets transiently, an intermediate node is able to respond to ARQ requests on behalf of the sender, which practically shortens the retransmission path. The rest of this paper is organized as follows. At first the benefits of the proposed strategy regarding perception quality and power consumption will be analysed in the next section. Then, the functional design will be laid out in Section 3, where caching behaviors and processing of ARQ requests are presented. Section 4 will introduce the algorithm for packet selection and forwarding. Before concluding the paper, we report the experimental results with a real-life testbed in Section 5.

2. RETRANSMISSION ANALYSIS

Suppose that video packets need to pass n hops to reach the receiver as illustrated in Fig. 1. As an intermediate node caches packets, it can reply to a NACK (Negative ACKnowledgment) message if some or all requested packet(s) is/are currently available. This section will evaluate the reduction of error rate and cumulated power consumption that the mechanism brings up.

1-4244-0411-8/06/$20.00 ©2006 IEEE.

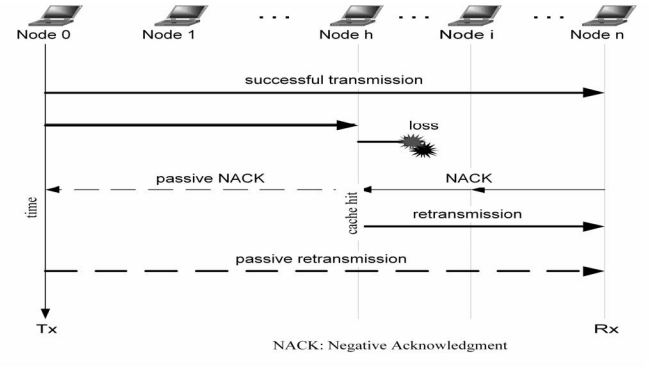


Fig. 1 Retransmission from an intermediate node.

2.1. Communication reliability

Assume that all the links along the path are loosely symmetric so that the back-route that conveys the NACK carrying ARQ request(s) is the same as the forth-route. If some requested packet is found cache-hit at node *h*, then the length of the retransmission round is only $2\times(n-h)$ hops, rather than $2\times n$ as in the case of *passive retransmission*. Thus, the round trip time (*rtt*) is approximately reduced by $\Delta rtt=\sum_{i=1}^{h}(d_i^f+d_i^b)$, where d_i^f and d_i^b are forth- and back-delay values of hop *i* (connecting node *i*-1 to node *i*), respectively.

Let's denote the bit error rate in forth- and back-direction as BER_i^f and BER_i^b, the length of the video packet and of the NACK as L_p and L_r, respectively. Then it can be inferred that the successful rate per retransmission round is increased by:

$$\Delta P_{success}^{re} = \frac{1}{\prod_{i=1}^{h}\left(1-BER_i^f\right)^{L_P}\times\left(1-BER_i^b\right)^{L_r}} \quad (1)$$

times, compared to the passive retransmission. The reduction of *rtt* also implicates more possible retransmission times before the requested packet misses deadline. The overall gain factor is therefore even higher:

$$\Delta P_{success} = \frac{1}{\prod_{i=1}^{h}\left(1-BER_i^f\right)^{L_P}\times\left(1-BER_i^b\right)^{L_r}}\times\frac{\sum_{i=1}^{n}d_i^f+d_i^b}{\sum_{i=h+1}^{n}d_i^f+d_i^b} \quad (2)$$

2.2. Power consumption

In each round of retransmission, node *i* receives the NACK from node *i*+1 and then transmits it to node *i*-1; later when the requested packet arrives, node *i* receives it from node *i*-1, and then transmits it downstream. So any intermediate node receives and transmits twice each.

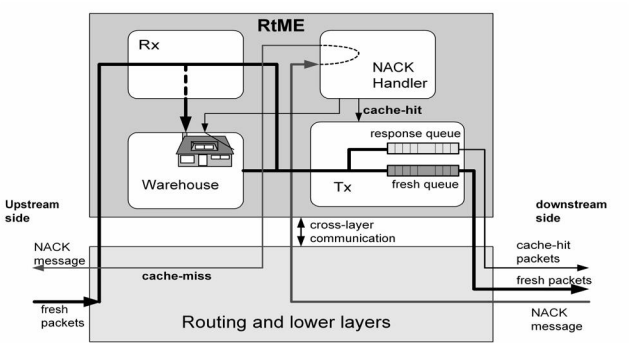


Fig. 2 Structure of RtME (Real-time Media Engine).

Obviously, the times of *transmit* and the times of *receive* are reduced each by: $\Delta N_{tx}=\Delta N_{rx}=2\times h$, compared to the case of passive retransmission.

Practically, both *receive* and *transmit* consume much more power than idle mode; as indicated in [5], for instance, *standby:receive:transmit* consumption ratio is 1:1.2:1.7. Thus, shortening retransmission rounds is always worth-doing. If the path consist of asymmetric links so that the forth- and the back-routes are not the same, higher priority should be given to the back-route that covers more nodes of the forth-route, while transferring NACKs.

3. ARCHITECTURE OF PROACTIVE NODES

To realize the responsiveness for adhoc nodes, we have designed a so-called *Real-time Media Engine* (RtME) that is a plugin-like module topping the routing layer, as depicted in Fig. 2. Once video packets arrive at a node containing RtME (hereafter called *proactive node*), they are taken for further processing. The main functions of RtME are caching video packets into the allocated memory called *warehouse*, processing NACK messages, and selecting the most appropriate packet for relaying each time the down hop is ready. These are executed in different entities: Rx, NACK handler, and Tx. As implemented in our testbed, they are the threads that are synchronized by a special *semaphore*.

3.1. Caching video packets

Every video packet incoming from upstream is called "fresh packet", though it might also be a retransmitted one generated elsewhere upstream. Because delay longer than a threshold D^{thres} is unacceptable, the maximum number of frames in the warehouse at any time is $N_c=D^{thres}\times R$, where *R* is the frame rate (practically less than 30 fps). To filter out obsolete frames, the Rx maintains two bounds. The first is the identification number of the latest arriving frame, called *fresh bound*. This number, denoted as *f*, is updated whenever a packet of a new frame arrives. The second bound, called *obsolete bound*, is defined as that of

the latest frame that is now missing deadline. Intuitively, this value can be calculated as $f - N_c$, and is updated synchronously with the fresh bound. Upon arriving at the node, a packet will be dropped if it is of a frame no later than the obsolete bound. Since every packet should not lie in the warehouse longer than D^{thres}, its maximum required capacity is estimated as:

$$C = B \times D^{thres} \quad (3)$$

where B is the bit rate of the video flow.

As stated in [6], D^{thres} is hereafter set to 500 milliseconds, so the maximum capacity required for the cache is one half of B, and N_c is less than 15. Practically, no one expects a bit rate of encoded video exceeding 1 Mbps in a wireless adhoc network, the required capacity is less than 62.5Kbytes. Obviously, allocation of such a limited RAM space is not unrealistic in today's adhoc devices. Once a fresh packet is accepted, it is enqueued into the designated queue called *fresh queue* as seen in Fig. 2, and one unit is added to the semaphore, notifying the Tx that another packet is available for forwarding.

3.2. Processing NACK messages

To minimize the overhead, successfully received packets are not acknowledged. A NACK message, whose format is shown in Fig. 3, is generated only when the receiver node detects lost packet(s). The first two bytes are to identify the just successfully received packet p, which helps the node update the current round trip time *rtt*, calculated as:

$$rtt = t_p^{now} - t_p^{ltx} \quad (4)$$

where t_p^{now} and t_p^{ltx} are respectively the current time and the time when packet p was last sent from this node. The updated *rtt* is used to decide whether a selected packet should be forwarded as there is enough time (presented later in Section 4.2).

If packet p is obsolete, the NACK is discarded decisively. Otherwise, a list of requested packets is interpreted based on the frame identification number of p, and on the remainder of the NACK, one byte per frame's packet(s). The "cache-hit" packets are then queued into the designated queue called *response queue* as seen in Fig. 2. The semaphore is also added one unit upon queuing a cache-hit packet. If there is any "cache-miss" packet, the node relays the NACK upstream, after cutting off its bytes corresponding to the cache-hit packets. So, the complexity is $O(N^{nack})$, where N^{nack} is the number of requested packets, and:

$$N^{nack} < D^{thres} \times R \times k \quad (5)$$

where k is the maximum number of packets per frame. This value pertains normally to I frames. In the experiments presented in Section 5, k is no greater than 5.

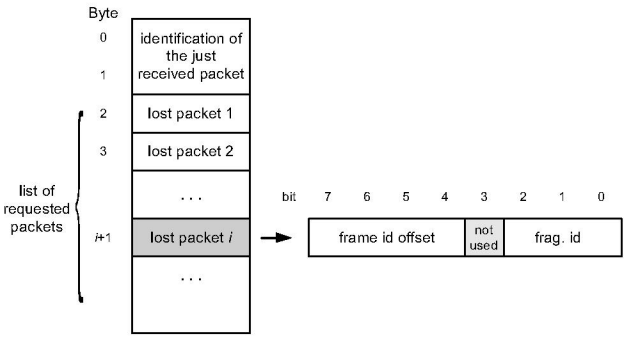


Fig. 3 Format of NACK messages.

4. PACKET SELECTION AND FORWARDING

To save the computation resource, the Tx thread is only awaked when the semaphore has its value positive. After dequeuing each packet, the Tx subtracts one unit from the semaphore's value.

4.1. Packet selection

In good propagation conditions, only the fresh queue has packets to forward. Otherwise, there are numerous algorithms for packet selection. We design an open algorithm with low complexity as seen in list 1. Note that, in addition to I, P, and B frames, each GoP contains a "header" frame, denoted as "H". Briefly speaking, the head-of-queue packets are read out to pointers `fh_pk_ptr` (pointing to "fresh packet") and `ch_pk_ptr` (pointing to "cache-hit packet"), the Tx then decides which one to return (as pointer `paket_to_send_ptr`). If both packets are of B, the cache-hit packet should be served since it has shorter remaining life-time; otherwise, if only one is of B, then the other should be of course selected. If the cache-hit packet is of H, I, or P, it will also be served unless it is of the earlier GoP while the fresh packet is also of H, I, or P. If one queue is empty, the other should be served if not empty else the Tx thread goes sleeping.

List 1 – A cycle of packet selection

```
paket_to_send_ptr = NULL;
/*read out the head-of-queue packets from the queues*/
fh_pk_ptr = head_packet(fresh_queue);
ch_pk_ptr = head_packet(response_queue);
/*decide which packet is selected for transmission*/
if (both queues are not empty)
{
 if (ch_pk_ptr->frame_type == B)
 {
  if (fh_pk_ptr->frame_type == B)
  {
     paket_to_send_ptr = ch_pk_ptr;
     Delete ch_pk_ptr from response queue;
  }
  else // fresh packet is of H, I, or P
  {
```

```
      paket_to_send_ptr = fr_pk_ptr;
      Delete ch_pk_ptr from fresh queue;
    }
  }
  else // cache-hit packet is of H, I, or P
  {
    if (fh_pk_ptr->GoP_id == ch_pk_ptr->GoP_id)
    {
      /* Both packets are of the same GoP */
      paket_to_send_ptr = ch_pk_ptr;
      Delete ch_pk_ptr from response queue;
    }
    else // cache-hit packet is of the previous GoP
    {
      if (fh_fr_ptr->frame_type != B)
      {
        paket_to_send_ptr = fh_pk_ptr;
        Delete ch_pk_ptr from fresh queue;
      }
      else
      {
        paket_to_send_ptr = ch_pk_ptr;
        Delete ch_pk_ptr from response queue;
      }
    }
  }
}
else //at least one of the queues is empty
{
  if (only fresh queue is not empty)
  {
    paket_to_send_ptr = fh_pk_ptr;
    Delete ch_pk_ptr from fresh queue;
  }
  else
  {
    if (only response queue is not empty)
    {
      paket_to_send_ptr = ch_pk_ptr;
      Delete ch_pk_ptr from response queue;
    }
  }
} /*end of pseudo-code*/
```

4.2. Forwarding decision

A selected packet is really forwarded only if there is enough time to reach the receiver, and it has not been sent within last *rtt* period. Specifically, both (6) and (7) must be satisfied; otherwise, it is dropped out and another cycle of packet selection is repeated, and so forth.

$$\begin{cases} D^{thres} - (t^{now} - t^{arrival}) > rtt/2 & (6) \\ t^{now} - t^{ltx} > rtt & (7) \end{cases}$$

where t^{now}, $t^{arrival}$, and t^{ltx} respectively stand for the current time, the arrival time and the time of the last *transmit* of the selected packet. Since *rtt*, as updated according to (4), depends on the goodness of the path, (6) makes the algorithm adaptive to channel conditions.

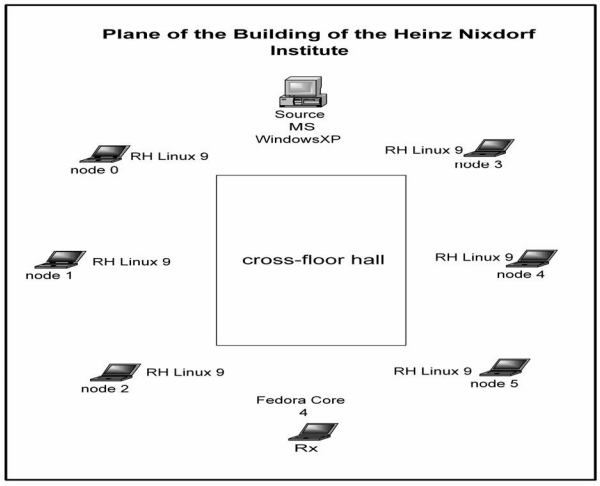


Fig. 4 Adhoc network layout.

The complexity of the whole process of selection and forwarding is very low. In the worst-case, it may have to scan all the queued packets, which is less than $N_c \times k$. Since each intermediate node does not rely on others to make decisions, the architecture is open and flexible.

5. EXPERIMENTS

We have been developing a testbed of 8 nodes (under Intel Pentium III and IV platforms), running RedHat Linux 9, Fedora Core 4, and WindowsXP, as shown in Fig. 4. The sender is a Desktop PC running WindowsXP and is equipped with Netgear USB WLAN card, whereas all other nodes are Laptops using Lucent Technologies PCMCIA WLAN cards for adhoc communication. Video traffic was delivered to the receiver node over two disjoint paths. Two experiments with video sequence *Akiyo* [7] were performed: one with our proactive framework, and the other with the conventional multi-path transport strategy. The MPEG4 CIF encoded sequence (300 video frames) was transmitted 30 times repeatedly to emulate real-time video. There were WLAN Access Points, student's laptops, and other adhoc networks in the place where the testbed was deployed.

5.1. Perception quality

The complete collection of PSNR (Peak Signal to Noise Ratio) statistics for the two cases is charted in Fig. 5. It is clear that the bold line (pertaining to the proactive approach) is above the dotted line most of the time. Indeed, the average PSNR of our framework (34.94dB) is 2.27dB better than the conventional case (32.67dB). Remarkably, our framework also reduces the time during which distortion is severe. Fig. 6 shows the episode-averaged PSNR, where the outperformance of our framework is more visible.

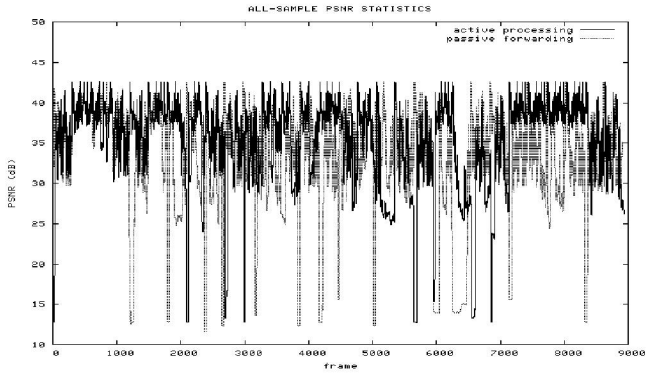
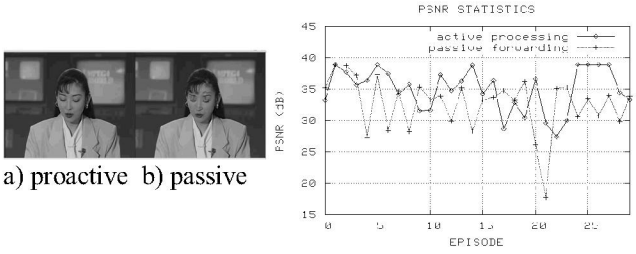


Fig. 5 PSNR of all frames.



a) proactive b) passive

Fig. 6 Episode-averaged PSNR statistics.

5.2. Power consumption

To measure power consumption more accurately, all the intermediate nodes ran their OS in *text mode*, turning off irrelevant processes/services, except daemon *olsrd* [8], which is a table-driven adhoc routing protocol. As seen in Fig. 7, in overall, the outperformance of our framework is noticeable; only node 1 consumed a little more power. We did not observe any reduction in node 3, but in other downstream nodes. This was probably due to random environment factors together with retransmission behaviors in MAC layer.

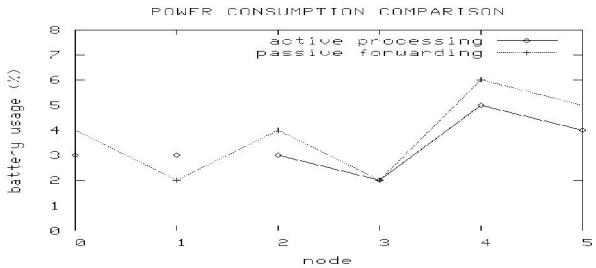


Fig. 7 Power consumption.

5.3. Cache usage

During the whole 30 episodes of communication, the peak warehouse fullness was only 24646 bytes and 10294 bytes measured in primary and extended paths, respectively. The episode-averaged caching statistics estimated on two paths are plotted in Fig. 8, further demonstrating that the memory allocation for caching is highly feasible.

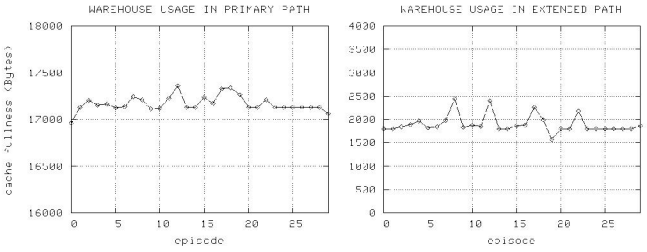


Fig. 8 Average cache usage.

6. CONCLUSION

We have presented a proactive architecture for intermediate adhoc nodes to efficiently relay real-time video packets. Both perception quality and power consumption have been considered as the architecture was designed. As adhoc networks are normally deployed in limited areas and each node serves very few users, if not single, at a time, the introduction of the framework is highly practical. As demonstrated, the framework is open to modifications and can inter-operate with other proposals from the research community. In our future work, the issue of energy-saving will be further addressed.

REFERENCES

[1] H. Gharavi and K. Ban, "Cross-layer feedback control for video communications via mobile ad-hoc networks," Proc. IEEE Vehicular Technology Conference, 2003.

[2] S. Mao, S. Lin, S. S. Panwar, Y. Wang, and EmbreCelebi, "Video transport over ad hoc networks: multistream coding with multipath transport," IEEE Journal On Selected Areas in Communications, pp.1721–1737, Dec 2003.

[3] A. Majumdar, D. G. Sachs, I. V. Kozintsev, K. Ramchandran, and M. M. Yeung, "Multicast and unicast real-time video streaming over Wireless LAN's," IEEE Trans. Circuits Syst. Video Technol., Vol. 12, pp. 524--534, June 2002.

[4] A. E. Al, T. Saadawi, and M. Lee, "Improving interactive video in ad-hoc networks using path diversity," Proc. IEEE International Conference on Mobile Ad-hoc and Sensor Systems, Oct 2004.

[5] A. M. Safwat, H. S. Hassanein, H. T. Mouftah, "Energy-aware routing in MANETs: analysis and enhancements," Proc. 5th ACM MSWiM, pp. 46-53, Sept 2002.

[6] X. Meng, H. Yang and S. Lu, "Application-oriented multimedia scheduling over lossy wireless networks," Proc. IEEE ICCCN, Oct 2002.

[7] J. Klaue, "YUV CIF video sequences," http://www.tkn.tu-berlin.de/research/evalvid/.

[8] Tonnesen, "Olsr daemon," http://www.olsr.org/.