Communication using Multiple Wireless Interfaces

Kameswari Chebrolu and Ramesh Rao
Department of ECE
University of California, San Diego

***Abstract*— With the emergence of different wireless technologies, a mobile terminal equipped with multiple interfaces can achieve a much higher bandwidth by aggregating the bandwidth offered by the individual networks. In this paper, we present a system based on Mobile IP that achieves the above objective. We will discuss in detail the architectural requirements and algorithms that are needed to support the above system.**

I. Introduction

A variety of wireless interfaces (Ricochet, GPRS, HDR, IS-95) are available these days to access internet content. The bandwidth offered by any of these interfaces is often not sufficient for demanding applications when used alone. A way to achieve higher bandwidth is to aggregate the bandwidth offered by the multiple interfaces by striping traffic across the interfaces. For example, if these different interfaces offered a bandwidth of 56, 128 and 144kbps individually, a mobile terminal equipped with them has at its disposal an aggregated bandwidth of 56+128+144 = 328kbps. Inverse multiplexing [1], Stripe protocol [2] are techniques that were proposed in the context of ISDN, ATM and analog dialups that do bandwidth aggregation over multiple (often similar) links. These implementations however assume stable link characteristics and are either infeasible in the present scenario or do not perform well over unstable wireless environments.

Ideally one would want the system with multiple interfaces to perform as well as a system with a single interface (same bandwidth). However it is difficult to achieve the same performance because most wireless networks display variable bandwidth, loss and latency. These variations usually cause packet reordering and can adversely affect the performance of any delay sensitive application. The situation tends to worsen when reliable transport protocols (like TCP) are used due to their extreme sensitivity to packet reordering and loss. When scheduling packets, it needs to be ensured that the interface bandwidths are properly utilized while minimizing packet delay, jitter and reordering. In this paper, we propose an algorithm *Earliest Delivery First* (EDF) that achieves the above objective by scheduling packets based on estimated delivery time on the different interfaces. Further to ensure that the different applications running on the mobile get their share of bandwidth, we propose a *Scheduling* algorithm that combines a fair queuing algorithm like WFQ (Weighted fair queuing) [3] with EDF. Another aspect to consider when using multiple interfaces is deciding which interface to use. Based on a suitably defined cost model, we propose an algorithm (*Interface Selector*) that selects the right interface/interfaces that minimize the *cost* of using the network while satisfying the bandwidth requirements of the applications.

The rest of the paper is organized as follows. In Section 2, we discuss the architectural details. The *Interface Selector* algorithm and the *Scheduling* algorithm are described in Section 3 and 4 respectively. We present some useful properties of EDF in Section 5. Section 6 discusses the simulation results and we finally conclude in section 7.

II. Architectural Details

Mobile IP [6] is a standard protocol that is used for supporting mobility in IP networks. We use a Mobile IP like infrastructure, where all the packets destined for the mobile pass through the home agent. Fig 1, shows the scenario. The mobile terminal is equipped with multiple interfaces and can use them simultaneously. Unlike standard Mobile IP, the mobile can now be associated with multiple care-of-addresses all at the same time. Also the home agent, instead of directing the packets it receives to a single address can now direct them to multiple addresses.

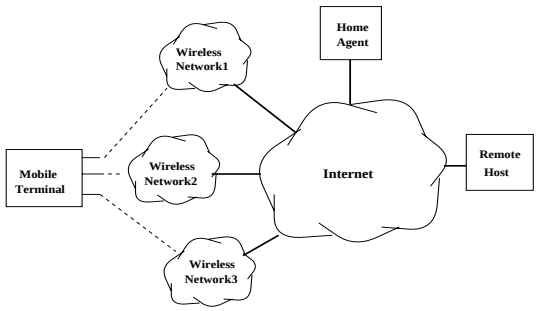

Fig. 1. General Scenario

An *Interface Selector* algorithm that picks the right interfaces based on application requirements is implemented on the mobile. A *Scheduling* algorithm that schedules packets on the different interfaces is implemented on both the mobile terminal and the home agent. Both the algorithms are explained in detail in the following sections. All the signaling needed when an interface is brought up/down or in case of handoff is as in Mobile IP. The only extension is that the home agent now needs to

be informed of the application's bandwidth requirements, delay suffered and bandwidth available on any interface to help it schedule packets. Due to varying nature of the bandwidth and delay, the mobile terminal needs to monitor these parameters periodically and inform the home agent of any necessary changes. In this paper, we will not focus on how the parameters are estimated and information conveyed but assume such a mechanism exists.

III. Interface Selector Algorithm

The services offered by any network can be associated with two parameters: the offered bandwidth and the cost one incurs for using the bandwidth. This cost will usually be a function of money paid, power consumed, user preferences etc. Such a cost model has been discussed in [5] in the context of handoff. In this paper, we will not discuss the cost model but assume such a model exists. Based on the model, we can order the networks in increasing order of cost per unit of bandwidth usage.

Any application on the mobile terminal usually requires certain quality of service which can be specified in terms of B_{min}, the minimum bandwidth required. Given this input, the algorithm now needs to pick up the least number of interfaces that minimizes the cost function while satisfying the bandwidth requirements of the applications. This can be achieved by ordering the networks in increasing order of costs, starting with the least cost network fill up the bandwidth of the networks till the bandwidth requirements are met. This ensures that this is the lowest cost one needs to pay for the required bandwidth. Once it is decided as to which interfaces to use, the interfaces that are not already up are brought up and the home agent is informed of the new changes so that it can schedule packets accordingly.

The above scenario is best illustrated by an example. Let a mobile terminal be equipped with three interfaces T1,T2 and T3, and let each of these technologies be associated with the following metrics: T1 (bandwidth = 1 unit, cost = 10 units/unit of bandwidth); T2 (bandwidth = 1 unit, cost = 20 units/unit of bandwidth); T3 (bandwidth = 2 unit, cost = 30 units/unit of bandwidth). The user wants to run three applications whose bandwidth requirements are 0.8, 0.5 and 0.2. Since the combined bandwidth is 1.5, the algorithm would pick T1 and T2, where the user would use 1 unit of bandwidth of T1 and 0.5 units of bandwidth of T2. The cost the user incurs is 20 units. Had he used any other combination of the interfaces, he would be paying a much higher cost.

One point that needs to be noted here is that when splitting connections onto multiple links, the aggregated bandwidth perceived by the connection is usually less than the sum of the bandwidths on the different links. Therefore the bandwidth input to the algorithm must be made slightly higher than B_{min}.

IV. Scheduling Algorithm

The scheduling algorithm needs to partition the traffic from multiple input queues (corresponding to each application) onto multiple output links (corresponding to each interface). We achieve the above objective by combining a fair queuing algorithm [3], [4] (which partitions traffic from multiple input queues onto a single output link) with a channel striping algorithm [2] (which partitions traffic from a single queue onto multiple links). The Stripe protocol [2] can be used as the channel striping algorithm but it was designed under the assumption that the links offer FIFO delivery. This results in a penalty in the form of synchronization between sender and receiver in case of packet loss and also large delay and jitter. We now describe our channel striping algorithm (EDF) and then explain how to combine WFQ(Weighted Fair Queuing) with EDF.

Since the packets generally follow different paths, they may not arrive in order at the receiver. The receiver usually must reconstruct the sending sequence. In our algorithm, we reduce the overhead one needs to pay for this reconstruction by taking the link(total path) characteristics into consideration when scheduling packets. Given the available bandwidth and average delay on each of the link, we know when a packet scheduled on this link is expected at the receiver. So, in our algorithm, we schedule packets based on which link delivers the packet the earliest. This way, one also reduces the delay and jitter experienced by the packets.

Each link l is associated with three quantities: a variable S_l, which is the time the link becomes available for the next transmission, D_l, the delay associated with the link and BW_l, the bandwidth of the link. If we denote by a_i, the arrival instance of the i^{th} packet and L_i, the size of the packet, we know that this packet when scheduled on link l would arrive at the receiver at R_i^l, where

$$R_i^l = MAX(a_i, S_l) + D_l + L_i/BW_l \quad (1)$$

The channel striping algorithm would schedule this packet on the link for which R_i^l is the minimum, S_l is then updated to $R_i^l - D_l$. When the packets are all of the same size and the delay on the links constant, it is easy to see that this scheduling achieves perfect ordering (no buffering needed). If the packets are of variable size, it is possible for packets to arrive out of order, in which case they can be buffered at the receiver and then delivered in sequence to higher layers. We provide some interesting properties of this algorithm in the next section.

The *Scheduling* algorithm is obtained by connecting the output of the WFQ to the input of EDF. When a packet arrives at the scheduler, it is determined as to which application it belongs and after calculating its departure time, it is placed in the application's queue. Each time, any of the links (say i) become

idle, the packet with the minimum departure time is selected (WFQ) and is scheduled on the link (say j) that delivers it at the earliest (EDF). Note that j need not be the same as i. This process of scheduling is repeated till a packet is scheduled on the idle link(i). It is possible that packets may not be scheduled on the idle link if there are not enough packets on the input queues. In which case, we erase the scheduling done previously and wait until enough packets arrive so that a packet can be scheduled on the idle link. It may be necessary under certain conditions to not use a portion of one of the link's bandwidth (as was the case in the example in the previous section where 0.5 units of bandwidth of $T2$ is to be left unused). To achieve this, we use a counter that keeps track of the idle and busy time of the link, if the busy time exceeds the idle time more than required, we advance the link's S_l till the ratio is as desired. This way, we ensure that the link is idle for the duration required.

Note that in reality the home agent does not have multiple interfaces to schedule packets on, rather it has only one output interface. The delay and bandwidth the home agent uses when scheduling packets are essentially the delay suffered by the packet between the home agent and the base station serving the mobile and the bandwidth available for the mobile on that interface. We are justified in making such an assumption as the wireless link is the bottleneck in any transmission since the wired links have very high data rates. The mobile of course has multiple interfaces to schedule packets on.

V. Theoretical Results

In this section, we state some properties of EDF without going into proofs due to restriction in space. The complete proofs can be found in [9]. In the analysis, we assume that the delay on the links is 0 and the first packet arrives at time 0. Note that long-term performance of EDF as such is not dependent on the delay on the links as long as they are constant. Different delays on the links just produce a transient effect and the results below hold after the transients die.

Let n be the number of links and let BW_i be the BW of link i. Then $BW_{max} = max\{BW_i\}$ and $BW_{min} = min\{BW_i\}$. Let weight of link i w_i be BW_i/BW_{min}. In the analysis, we define jitter as the maximum difference in delay experienced by two consecutive packets. We say an input queue is backlogged with N packets if whenever a link becomes idle, there is either a packet in the queue that can be scheduled or all the N packets have already been scheduled.

Theorem 1: (Fairness Measure) Given N packets to transmit, the difference between the normalized bits allocated to any two links when the input queue is backlogged is bounded as follows:

$$\left| \frac{Sent_i}{w_i} - \frac{Sent_j}{w_j} \right| \leq L_{max}, \tag{2}$$

where, $Sent_i$ is the number of bits allocated to link i and L_{max} is the maximum packet length. Note that this upper bound is $2L_{max}$ for Stripe protocol.

Theorem 2: (Time Bound) Given N packets to transmit, the time taken to finish the transfer when the input queue is backlogged is upper bounded as follows:

$$\frac{\sum_{j=1}^{N} l_j}{\sum_i BW_i} + \frac{(n-1) * L_{max}}{\sum_i BW_i}, \tag{3}$$

where l_j is length of packet j. Note that the second term is in essence the additional time taken when multiple links are used in place of a single link whose BW is the sum of the BWs of the multiple links.

Theorem 3: (Jitter Bound) The jitter experienced by the packets at the higher layers when the packets are buffered at the receiver is upper bounded by L_{max}/BW_{max}. The jitter experienced by packets as received before buffering is upper bounded by L_{max}/BW_{min}. Note that this upper bound even with buffering is L_{max}/BW_{min} for Stripe protocol.

Theorem 4: Buffering Required: The buffer size needed to deliver packets in order is upper bounded by $(n-1) * L_{max}$.

VI. Simulation Results

The performance of the system depends heavily on how well channel striping works. Due to restriction in space, we furnish only important results and explain in passing the other results we obtained. We refer to [9] for more detailed results.

We have examined the performance of EDF using a trace driven simulation. Fig. 2 shows the set up. We have considered three links, the bandwidth and delay on the links is as in fig 2. In this paper we focus on real-time applications where jitter and delay are important parameters. No transport protocol is considered though we wish to study in future the effect of TCP on the system. The traffic flow is from sender to the mobile receiver. The sender generates video frames that are divided into packets and forwarded to the home agent. The home agent then schedules the packets onto the multiple links (the three links corresponding to each interface are the different paths taken by the packets). At the mobile receiver the packets are buffered and delivered in order to higher layers. The higher layers then reconstruct the frames. On the sender side, we used a frame size trace (Mr.Bean, a cable talk show) from MPEG-1 encoded video sequences. The encoder input is 384x288 pel. The capture rate is 25 frames/sec. The mean and peak bit rate of the trace are 440kbps and 1760kbps respectively. We refer to [7], [8] for further details about the trace.

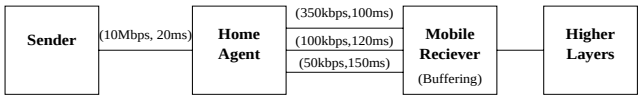

Fig. 2. Set Up of the system

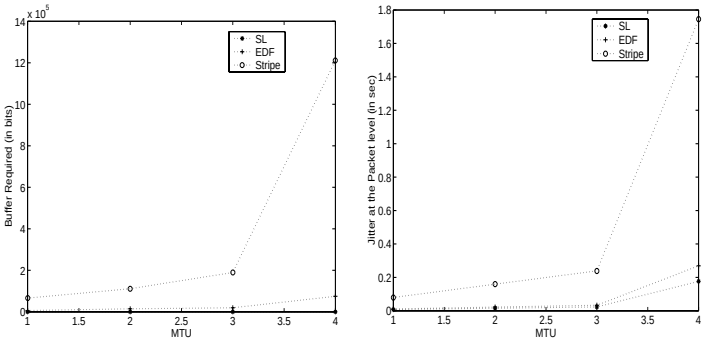

Fig. 3. Buffering and Jitter at the packet level

We compare EDF with two other algorithms: a Single Link Algorithm (SL), where the multiple links between the home agent and the mobile are replaced with a single link whose BW is sum of the BWs of the multiple links. The delay on this link is maximum of the delays on the multiple links. This comparison should give a measure of how well channel striping works. The closer it mimics the SL, the better the channel striping algorithm. We also compare our algorithm with the Stripe protocol [2]. We used the following metrics to evaluate the performance of the algorithms: Overall throughput(thr) measured at the home agent, time taken (time) to finish transfer of packets measured at the mobile, buffering required (buff), average delay (avg), maximum delay (max) and jitter (jit) experienced by the packets, average delay, maximum delay and jitter experienced by the frames and the frame throughput (fr_thr). All these parameters are measured at the higher layer at the mobile.

Table I shows the results of our simulation when the delay on the links is assumed constant and a maximum transmission unit (MTU) of 12000 bits is used. Note that the performance of EDF is as good as SL. The Stripe system on the other hand has much larger buffering requirements and hence more reordering (9.6 times ours) and a large jitter at the packet level (7.2 times ours). Even the jitter at the frame level is considerably more.

MTU size variation: We have also studied the performance of the system under different MTU sizes (gives a measure of the algorithm's response to variability in packet sizes). We considered four MTU sizes: 1) 4000, 2) 8000, 3) 12000 and 4) infinity (no restriction on MTU). We observed that as the MTU size was increased, EDF still closely followed the SL system but the Stripe protocol performance got worse. The reason being that the worst-case scenario: no restriction on packet size corresponds to a large variation in packet sizes. Such large variations are not tolerated well by Stripe protocol. The buffering required and jitter at the packet level were the parameters that showed the most sensitivity to MTU variation. Fig. 3 shows the variation of these parameters as the MTU size is varied. For other parameters, sensitivity with respect to MTU size variation was more in case of Stripe than with SL or EDF.

Delay Variation: We have also evaluated the performance of the system under varying delay on the links. The delay variation was assumed to be truncated Gaussian with mean as given by the above values. Though we have not looked at packet loss in this paper, if a retransmission policy is in place, it can be accounted for by the large delay experienced by some of the packets. We have considered two cases here, a small delay variation and a large delay variation. For small delay variation, the standard deviation was assumed to be 10ms on the multiple links and 5ms on the link between sender and home-agent. For large delay variation, the standard deviation was assumed to be 50ms on the multiple links and 10ms on the link between sender and home-agent. The performance of all the systems degraded as the variation increased but the relative performance of all the systems was as under constant delay i.e EDF continued to perform as well as SL with Stripe performing worse than the two. Table II shows the performance of the systems under large delay variation and no restriction on MTU size. For any MTU size, the packet jitter with delay variation has taken on a much higher value for SL and EDF than under constant delay. The reason for this is that the delay variation on the link between sender and home agent introduced severe reordering at the home agent.

VII. Conclusion

In this paper, we have described a system that aggregates the bandwidth offered by different wireless technologies. With demand for higher bandwidths in the wireless domain, such a system improves the quality of connection as perceived by the end user. We have explained in detail two algorithms: the *Interface Selector* algorithm and a *Scheduling* algorithm that are needed to achieve the above objective. We also showed a way of combining fair queuing algorithms with channel striping algorithms to schedule multiple flows on multiple links. We evaluated the performance of the proposed system and observed that under the assumptions we made, the system performs as well as a single high-speed data link.

References

[1] J.Duncanson, "Inverse Multiplexing", IEEE Communications, pp. 34-41, April 1994.

[2] Hari Adiseshu, George Varghese, and Guru Parulkar, "An Architecture for Packet-Striping Protocols," ACM Transactions on Computer Systems, Vol. 17, No. 4, November 1999, pp. 249-287.

[3] A. Demers, S. Keshav and S. Shenker, "Analysis and Simulation of a Fair Queuing Algorithm," in Proc. SIGCOMM'89, Vol. 19, No. 4, September 1998, pp. 1-12.

[4] M. Shreedhar and George Varghese, "Efficient Fair Queuing Using Deficit Round-Robin," ACM Transactions on Networking, Vol. 4, No. 3, June 1996, pp. 375-385.

[5] Helen J. Wang, Randy H. Katz, and Jochen Giese, "Policy-Enabled Handoffs Across Heterogeneous Wireless Networks", in Proc.of WMCSA, February 1999.

[6] Charles E. Perkins, "Mobile IP," IEEE Communications, pp. 84-99, May 1997.

[7] Oliver Rose, "Statistical properties of MPEG video traffic and their impact on traffic modeling in ATM systems," Proceedings of the 20th Annual Conference on Local Computer Networks, Minneapolis, MN, 1995, pp. 397-406.

[8] Location of MPEG-1 video traces, "http://nero.informatik.uni-wuerzburg.de/MPEG/traces."

[9] Kameswari Chebrolu and Ramesh Rao, " Theoretical Analysis and Simulation Results of Earliest Delivery First(EDF) Channel Striping Algorithm," Technical Report http://www-cwc.ucsd.edu/∼chebrolu.

TABLE I

CONSTANT DELAY, MTU = 12000 BITS

Algorithm	Thr (kbps)	time (sec)	buff (bits)	Packet Level avg (sec)	Packet Level max (sec)	Packet Level jit (ms)	Frame Level avg (sec)	Frame Level max (sec)	Frame Level jit (ms)	fr_thr (frames/sec)
SL	438.05	200.142	0	5.369	15.524	22.8	4.826	15.532	176.4	25.023
EDF	438.04	200.096	19672	5.354	15.508	33.1	4.808	15.515	196.7	25.028
Stripe	437.83	200.212	189552	5.432	15.623	238.8	4.886	15.623	279.8	25.013

TABLE II

LARGE DELAY VARIATION, MTU = NO RESTRICTION

Algorithm	Thr (kbps)	time (sec)	buff (bits)	Packet Level avg (sec)	Packet Level max (sec)	Packet Level jit (ms)	Frame Level avg (sec)	Frame Level max (sec)	Frame Level jit (ms)	fr_thr (frames/sec)
SL	438.05	200.140	42536	4.836	15.556	222.5	4.836	15.556	222.5	25.019
EDF	438.03	200.114	71944	4.877	15.701	317.5	4.877	15.701	317.5	25.033
Stripe	435.62	201.271	1294824	5.611	17.036	1760.2	5.611	17.036	1760.2	24.887