1998 International Computer Symposium
Workshop on Computer Networks, Internet, and Multimedia
December 17-19, 1998, N.C.K.U., Tainan, Taiwan, R.O.C.

Channel-Combine Strategies for On-Demand Multicast on Mobile Wireless Networks*

Kuo-Wei Chen, Ming-Hour Yang, Yu-Chee Tseng, and Jang-Ping Sheu
Department of Computer Science and Information Engineering
National Central University
Chung-Li, 32054, Taiwan

Abstract

This paper considers a communication problem in a closed area, such as stock market, sport stadium, or showroom, where people may move around and dynamically request for a piece of data. PDAs or hand-held PCs downloading the data through wireless networks is a promising solution to such scenarios. One naive solution is to allocate for each request a separate channel, but this will require a lot of channels if the demand is high. One way to relieve this problem is to adjust the speeds that mobile hosts receive data such that if two mobile hosts are on the same data point, one of them can be handoff to listen to the channel of the other. In this way, the channel of the former mobile host can be released. However, this should be done only if the pre-specified QoS requirement is not violated. In this paper, we develop three such channel-combine strategies, called Binary, Shortest-Distance-First, *and* Partition. *Simulation and experimental results are also presented.*

1 Introduction

One major breakthrough in computer communication technologies recently is the extension of transmission media from wired networks to wireless networks. Wireless products, ranging from LAN, MAN, to WAN, are available commercially, such as WaveLAN by Lucent, AIRLAN by Solectek, BreezeNET by BreezeCOM, RangeLAN and RangeLINK by Proxim, AirLink Bridge by Cylink, ARDIS, CDPD, DECT, and GSM [4, 5]. This, together with the popularity of small, light-weight, economic hand-held laptops, palmtops, and PDAs, has made *mobile computing* (or *nomadic computing*) possible [1, 8]. Mobility has created a new dimension of thinking in both computation and communication societies [2]. Users can carry computers (or *mobile hosts*) moving from locations to locations, while still remaining in touch with the rest of the world.

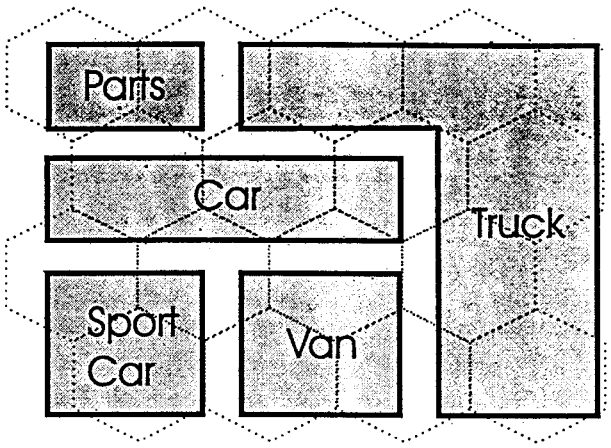

Figure 1. An example of a cellular wireless network for a car showroom.

In this paper, we consider a cellular network in a physically closed area, such as stock market, sport stadium, showroom, convention center, or horse-racing field. See Fig. 1 for an illustration. Users may be equipped with portable devices, roaming inside the cellular network and dynamically requesting data. The possible data contents may include stock market quotes, histories of stocks, results of games, information of football teams, statistics of racing horses, weather conditions, etc. Apparently, these data contents may change dynamically too. Also, these data may be pre-sorted into different categories, so users can request the pieces of data of interest.

PDAs or hand-held PCs downloading these data through wireless networks is a promising solution to

*This work is supported by the National Science Council of the Republic of China under Grant # NSC88-2213-E-008-014 and #NSC88-2213-E-008-027. Email: {yctseng, sheujp}@csie.ncu.edu.tw.

such scenarios. One naive solution is to assign for each request a separate channel, but this will require a lot of channels, especially in crowded areas. As many users may be interested in the same piece of data, the traffic should be *multicast* in nature. An alternative is to use a channel to broadcast the data periodically. This may still be unsatisfactory as a newly arrived user must wait till the beginning of the next period to start receiving information, especially when the period is long. Improvement can be obtained if we use multiple channels each periodically broadcasting the same piece of data but at a different point of time (e.g., [6, 7]). A mobile host only needs to wait till the next nearest period. This should reduce the average waiting time, but could be a waste of channels when there are fewer users than the number of channels.

In this paper, we consider to use an *on-demand* approach, where data is sent only on users' requests. We assume that each piece of data consists of a sequence of packets that must be delivered in that order. Multiple channels are used to support this data service. The packets must be delivered to each user following some quality-of-service (QoS) constraint. If two mobile hosts are receiving different packets on the sequence, it is possible to adjust (by slowing or accelerating) the speeds of their channels. Once the two mobile hosts are on the same data packet, one of them can be handoff to the channel of the other. In this way, the former mobile host's channel can be released. Newly arrived users and handoff users may benefit from this as there is more chance to find vacant channels. However, this should be done only if the pre-specified QoS constraint is not violated. Under such assumption, how to schedule the combining of channels to increase system performance becomes an interesting problem, which we call the *channel-combine* problem.

The channel-combine problem is also signified by the possibility of user mobility and the unreliability of wireless communication. Consider the simplest approach where the system assigns one channel per cell to periodically broadcast a piece of data. When a mobile host experiences handoff or temporary link breakage, it may miss some packets. This would force the mobile host to listen to the next broadcast period, which is very unfavorable for highly mobile users.

In this paper, we develop three channel-combine strategies, called *Binary*, *Shortest-Distance-First*, and *Partition*. The Binary strategy is the simplest but most inefficient. The Shortest-Distance-First strategy starts with combining the two channels that are on the closest packets, and then repeats the process recursively. The Partition strategy first divides the channels into groups according to their proximity in the packet sequence, and then deals with each group independently using either of the above two strategies. Through simulations, we evaluate these strategies on their channel throughput and forced dropping rate at various arrival rates, handoff rates, and QoS requirements.

Multicasts based on other models in wireless networks have been proposed. In [3], the bandwidth of a cell is divided based on the current user groups in this cell so as to maximize the throughput. However, it is assumed in [3] that the bandwidth is infinitely divisible, as opposed to the concept of channel (with a fixed, un-divisible bandwidth such as those in TDMA and FDMA) used in this channel. In extreme cases, the scheme may assign all bandwidth to a certain group of users. Reference [10] considers the scheduling of broadcasting data packets to users. Two schemes call pull-based and push-based are proposed with the goal of minimizing the waiting time experienced by users. In both works, the notion of QoS is not considered.

The rest of this paper is organized as follows. Preliminaries are in Section 2. Section 3 presents our channel-combine strategies. Simulation results are in Section 4. Conclusions are drawn in Section 5.

2 Preliminaries

In this section, we first present some channel management examples. We will discuss the strength and weakness of these methods. Through these examples, we motivate the channel-combine problem. We then define the problem formally.

2.1 Some Channel Management Examples

Example 1 *(Unicast)* In applications such as VoD (Video-on-Demand), data does not change frequently but might be large in volume. When a mobile host newly arrives at or handoffs to a cell, the simplest solution is to allocate for the user a new channel, provided that there are vacant ones; otherwise, the host is blocked until a new channel is released.

Let λ be the arrival rate (including new/handoff users) to a cell. Suppose that it takes T time units to send the data and the system has c channels. Then the average number of channels used will be $\min\{c, \lambda T\}$. The average waiting time can be found using the $M/M/m$ model [9] (an m-server queuing system). Such a system is not stable if λ is too high. □

Example 2 *(Periodic-Single)* To make the system stable independent of the arrival rate λ, the simplest solution is to make use of the broadcast nature of wireless transmission. Using one channel per cell to broadcast the data periodically would be sufficient. All users, on needing the data, must wait till the beginning of the

next period to start with. The average waiting time is $T/2$.

However, the scheme may not be appropriate for a mobile system. When a mobile host is handoff to a new cell, the new cell must be broadcasting the same packet; otherwise, the data stream is broken and the mobile host must restart from the next period. This places a strong demand that the handoff latency must be negligible. In a similar scenario, if a host experiences temporary link breakage, then it has to restart too. □

Example 3 *(Periodic-Multiple)* One way to improve the previous method is to allocate c channels each broadcasting the data periodically, but the period is shifted by T/c time for each channel. When a mobile needs the data, it only waits for the next nearest period, thus reducing the average waiting time to $T/2c$. □

Next, we demonstrate a channel-combine example.

Example 4 *(Channel-Combine)* In Fig. 2(a), it shows two mobile hosts x and y requesting the same piece of data using two channels. At time 0, host x just starts with the 0-th packet, while host y is already at the 100-th packet. Suppose that a channel can deliver 10 packets per time unit, and the QoS specifies that a minimum of 8 packets must be delivered per time unit to each host. If we send packets to x and y at speeds of 10 and 8 packets per time unit, respectively, then after 10 time units they will receive up to the 100-th and 180-th packets, respectively, as shown in Fig. 2(b). By keeping the same speeds, both hosts will receive up to the 500th packets after 50 time units, at what time one channel can be released and the other channel can send packets in the full speed of 10 packets per time unit, as shown in Fig. 2(c). The whole process of x catching up with y is shown in Fig. 2(d).

In Fig. 2(e), we show another alternative by delivering 10 and 9 packets per time unit to host x and y, respectively. If so, the two channels can be combined at time 100 at the 1000-th packet.

To compare these two approaches, consider the time interval from 0 to 100. The first approach uses an average of 1.5 channels in this interval, while the second 2 channels. At time 100, for both approaches, the system has delivered 1000 packets to x and 900 packets to y. So apparently the first approach delivers more packets to mobile hosts per channel per time unit than the second one. □

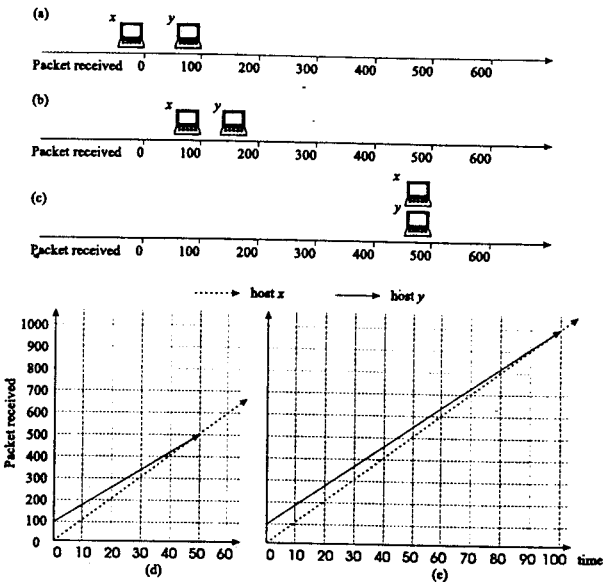

Figure 2. (a)-(d) show a channel-combine example if we send packets to x and y at speeds of 10 and 8 packets per time unit, respectively; (e) shows the case if 9, instead of 8, packets are sent to y per time unit.

2.2 The Channel-Combine Problem

We now formally define the problem. There is a data stream of m packets that should be delivered in that order on users' requests. We assume a cellular network of a number of cells. Within each cell, n channels are allocated to support this on-demand service. Mobile hosts in the system can move around and request this data service at any time. The request arrival rate is λ_a per second per cell. Each user, while receiving a data service, has a handoff rate of λ_h. We assume that the network already supports multicast in that more than one mobile host can listen to the same channel at the same time.

We assume that a channel can deliver s packets per second. Also, the QoS requires that each user requesting this service must receive the data at a minimum speed of s' packets per second, where $s' \leq s$, i.e., the actual delivery speed could fall between s' and s, inclusively. Alternatively, we may ignore the absolute values of s and s' and state that "the QoS tolerates a *slowdown* of $\delta = 1 - \frac{s'}{s}$." For instance, if $s = 10$ and $s' = 9$, then the tolerable slowdown is 10%.

In this paper, we only consider each cell independently. Given a cell in which there are k groups $g_1, g_2, \ldots, g_k$ of users each receiving up to the d_1-th, d_2-th, ..., d_k-th packets, respectively, the *channel-combine* problem is to determine a schedule of the speeds of

channels, without violating the QoS requirement, so as to vacate more channels to increase the performance of the system. The performance evaluation criteria may include:

- the average number of channels used during a time interval,
- the *channel throughput*, which is defined to be the total number of packets delivered to users divided by the channel-time product during a time interval, where *channel-time product* is defined to be the integral of the number of channels used over the time interval, and
- the *forced dropping probability* due to channel unavailability for handoff/new users.

To support such a system, the network must support a management channel to notify mobile hosts which channels to listen to at what time. In particular, when mobile hosts belonging to two channels are merged, the mobile hosts must know which channel to listen to next.

Finally, we comment on how the concept of slowdown can be extended. In some real-time services such as VoD, the system should deliver, say, 30 frames per second to a user, but it is tolerable to drop up to 2 frames among the 30 frames without users' notice. In this case, we can imagine that we use a channel speed of $s = 28$ packets to deliver $s' = 30$ packets (2 of which are dropped) per second to a user. Hence, we regard as if the system can tolerate a *speedup* of $\frac{s'}{s} - 1$. In fact, the notions of slowdown and speedup are equivalent, if we formulate this problem by specifying that each channel can transmit in a speed ranging between s and s'. With such understanding and without loss of generality, in the rest of this paper we will only consider the slowdown case.

3 The Channel-Combine Strategies

Recall that we are given k groups $g_1, g_2, \ldots, g_k$ of users in a cell each receiving up to the d_1-th, d_2-th, ..., d_k-th packet, respectively. Without loss of generality, assume $d_1 < d_2 < \cdots < d_k$.

To specify how channels are combined, we associate with each g_i an extension $g_i.target$, which is another group in front of g_i that g_i tries to catch up with. However, it is possible that $g_i.target = g_i$, which means that packets should be delivered to g_i at the slowest speed of s', in order for other groups to catch with it. On the contrary, for any g_i such that $g_i.target \neq g_i$, packets must be delivered to it at the fastest speed of s. When two group g_i and $g_j, i < j$, are on the same data packet, users in g_j should switch to the channel of g_i and the channel of g_j is released. After the merge, the target

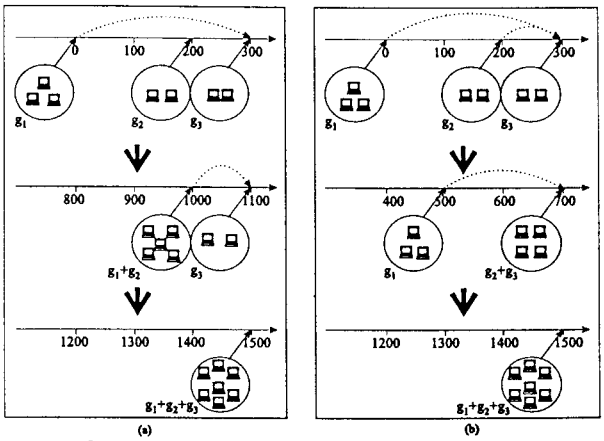

Figure 3. Channel-combine examples.

of the union $g_i \cup g_j$ is assumed to be $g_i.target$. Then, packets may be delivered to $g_i \cup g_j$ at a slow speed of s' if $g_i.target = g_j$, or at a full speed of s otherwise. These rules remain effective except when all groups are merged into one, in which case $g_i.target = g_j$, but it makes no sense to deliver packets at a slow speed of s' as there is no other group trying to catch up with this one. So a full speed of s is applied instead.

For instance, in Fig. 3, there are three groups g_1, g_2, and g_3 each at the 0-th, 200-th, and 300-th packet, respectively. Each dotted arrow leads a group to its target. A group without such arrow has a self-pointing target. Suppose $s = 100$ and $s' = 80$. In Fig. 3(a), we have $g_1.target = g_3$, $g_2.target = g_2$, and $g_3.target = g_3$. So in the beginning g_1 is of speed s and g_2 and g_3 are of speed s'. At the 1000-th packet, g_1 and g_2 will be merged. Then we will set $(g_1 \cup g_2).target = g_3$. A speed of s will be used by $g_1 \cup g_2$, leading to the merge with g_3 at the 1500-th packet. After that, a full speed of s will be used.

In Fig. 3(b), we let $g_1.target = g_2.target = g_3.target = g_3$. So in the beginning, g_1 and g_2 will be of speed s, and g_3 of speed s'. Groups g_2 and g_3 will merge first. Then we will set $(g_2 \cup g_3).target = g_3$, so a speed of s' will be used by $g_2 \cup g_3$, until g_1 catching up with $g_2 \cup g_3$.

3.1 Binary Strategy

In the binary strategy, we first try to combine adjacent groups (i.e., group g_1 with g_2, group g_3 with g_4, and so on). Then we repeat the process recursively to combine them. In the following, we use $[x, y, \ldots]$ to denote a sequence, where x and y are items. Note that x and y may in turn be sequences. Also, we use $x|y$ to denote the concatenation of x and y, and $|x|$ the number of items in sequence x.

Algorithm Binary;

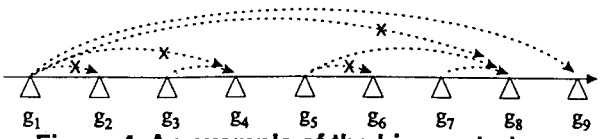

Figure 4. An example of the binary strategy.

1) Let the linked list $L = [[g_1], [g_2], \ldots, [g_k]]$;
2) **while** ($|L| > 1$) **do**
   a) Let $L' = [\,]$ be an empty linked list;
   b) **repeat**
      if $|L| > 1$ **then**
         Remove the first two items, say x and y, from L;
         Let the target of the first group in x equal the last group in y;
         Append $x|y$ to the end of L';
      else append the only item in L to the end of L';
      end if;
   until ($L = [\,]$);
   c) $L = L'$;

end do;

Fig. 4 shows an example in a cell with nine groups $g_1..g_9$. Each dotted arrow leads a group to the target group to chase. A group without such an arrow has a self-pointing target. The arrows with the crossed-out symbol "x" are those that were established in earlier stages of the algorithm but were replaced later by other arrows.

3.2 Shortest-Distance-First (SDF) Strategy

The binary strategy decides the combining targets based only on the order among channels. Its merit is simplicity. However, the distances between channels are not taken into consideration. In the Shortest-Distance-First (SDF) strategy, two channels of the minimal distance in the packet sequence will be merged first. This will reduce the number of groups by one. Then, based on these newly constructed groups, we repeatedly merge channels in a recursive manner, until the number of groups is reduced to one. This places a need to determine how much time a list of channels can be merged into one, so we have the following definition.

Definition 1 *Given a sequence of groups* $x = [g_i, g_{i+1}, \ldots, g_{i+j}]$, *we define* $dist(x) = d_{i+j} - d_i$, *i.e., the difference between the first and the last groups' packet positions.*

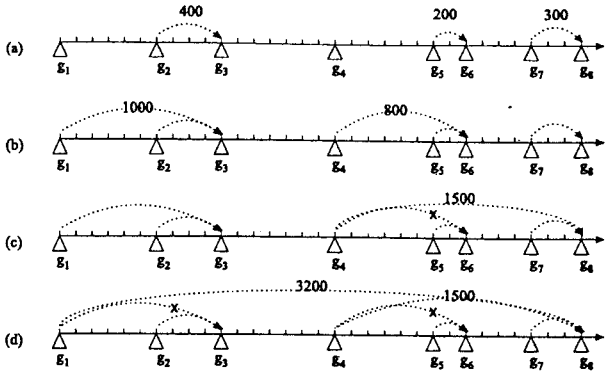

Figure 5. An example of the Shortest-Distance-First strategy. Each unit in the packet sequence represents 100 packets.

We formally develop the channel-combine strategy below.

Algorithm SDF();
1) Let $L = [[g_1], [g_2], \ldots, [g_k]]$;
2) **while** ($|L| > 1$) **do**
   a) Let x and y be the adjacent items in L such that $dist(x|y)$ is minimum among all adjacent x and y in L;
   b) Let the target of the first group in x be the last group in y;
   c) Replace the x and y in L by $x|y$;

end do;

Fig. 5 shows an example how this strategy works. In (a), the distance from g_5 to g_6 is the shortest, so $g_5.target$ is set to g_6. The next two shortest distances are g_7 to g_8 and g_2 to g_3, so $g_7.target = g_8$ and $g_2.target = g_3$. In (b), the next shortest distance is determined to be from g_4 to the union $g_5 \cup g_6$ (recall Definition 1). Similarly, the next shortest distance is from g_1 to $g_2 \cup g_3$. In (c), the next shortest distance is from $g_4 \cup g_5 \cup g_6$ to $g_7 \cup g_8$, so the value of $g_4.target$ is updated to g_8. Finally, in (d), only two groups remain and there is only one way to merge them, so $g_1.target$ is updated to g_8.

Lemma 1 *When there are only three groups, the SDF strategy always gives the best throughput.*

One would wonder if this strategy is optimal in all cases. Unfortunately, this is not true. Fig. 6 shows a counter-example with four groups in a cell. The solution in Fig. 6(a) is obtained from the SDF strategy,

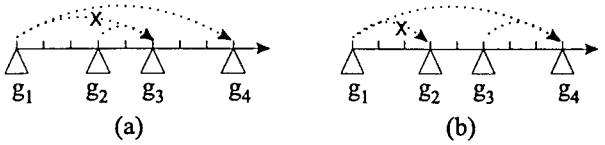

Figure 6. A counterexample in which the Binary strategy will give a better throughput than the SDF strategy. Each unit in the axis represents 100 packets.

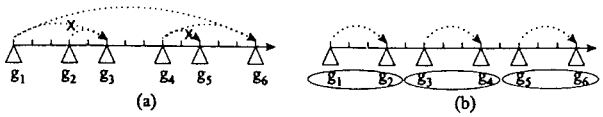

Figure 7. Comparison of the SDF and the Partition strategies.

which gives a channel-time product of $(4 * 200 + 3 * 300 + 2 * 300)/(s - s') = 2300/(s - s')$. Using the Binary strategy, as shown in Fig. 6(b), the channel-time product will be $(4*300+5*200)/(s-s') = 2200/(s-s')$, which is better.

3.3 Partition Strategy

In the previous strategy, the termination time of a group is not considered. For instance, if g_i decides to chase g_{i+1}, but g_{i+1} is already receiving the last few packets of the sequence, then g_i may not be able to catch up with g_{i+1} before g_{i+1} completes. In the Partition strategy, we propose to use a pre-determined constant T to partition the user groups into a number of sets, such that in each set no two groups are at data positions of a difference more than T packets. After the partitioning, we can apply any strategy (such as Binary or SDF) on each of the set independently to determine a channel-combine solution.

Algorithm *Partition*();

1) Let $L = [g_1, g_2, \ldots, g_k]$;
2) **while** $(L \neq [\,])$ **do**

   Partition L into two subsequences L_1 and L_2 such that $L = L_1|L_2$, $dist(L_1) \leq T$, and L_1 is longest;

   Call the Binary or SDF strategy with L_1 as the input;

   $L = L_2$;

end do;

It is possible to use the constant T to enforce the system to vacate a certain number of channels within some time interval. For instance, consider the example in Fig. 7. Part (a) shows the result obtained by the SDF strategy: after $\frac{200}{s-s'}$ time units, two channels will be released, and more channels can be released after another $\frac{300}{s-s'}$ more time units. In part (b), we use the Partition strategy by setting $T = 300$ packets. Three sets of groups are obtained (shown in circles). After applying the Binary or SDF strategy on each set, the result is three channels being released after $\frac{300}{s-s'}$ time units.

The above example demonstrates that if we appropriately set the value of T, we can predict the number of channels to be released after $\frac{T}{s-s'}$ time units. This is a favorable property if the system is in urgent need of free channels within a certain time. This could also be useful in a highly mobile environment, where newly arriving mobile hosts may disturb the channel-combine solution that was decided earlier, thus making the system unstable.

4 Simulation Experiments

4.1 Simulation Model

To compare the three channel-combine strategies, we simulated a system containing 16 cells which form a 4×4 mesh with wrap-around links (like a torus). Mobile hosts may move around in east, west, south, or north direction. When touching the boundary of the mesh, a mobile host can follow the wrap-around links to arrive at the cell at the other end on the same row/column. Each cell has $n = 20$ channels to support this data service. New mobile hosts arrive at each cell to request the data service with a rate of λ_a. While requesting the data, a host has a handoff rate of λ_h, with equal probability on each direction.

When a mobile host newly arrives at or handoffs to a cell, we assume that there is a setup latency of 10ms (involving all hardware, software, and communication overheads). However, if the current cell does not have an empty channel by 10ms, the mobile host will drop the connection. We call this a *forced drop*. Otherwise, a mobile host, once setting up, does not drop from the service until completion. The parameters used in our simulation are summarized in Table 1.

4.2 Experimental Results

We vary λ_h, λ_h, and δ, and observe how the proposed strategies perform in terms of channel throughput and forced dropping probability.

A) Comparison of Channel-Combine Strategies: Fig. 8 gives a comparison of our three strategies on channel throughput at various λ_a and δ values with

Table 1. Parameters Used in the Simulation.

PARAMETERS	SYMBOL	VALUE
Data Size (packets)	m	100000
Channel Speed (packets/sec)	s	10000
Initial Setup Time (msec)	T_s	10
Channels per Cell	n	20
Arrival Rate (calls/sec)	λ_a	≤ 20
Handoff rate (times/sec/user)	λ_h	≤ 0.02
Slowdown Ratio	δ	$1 \sim 20\%$

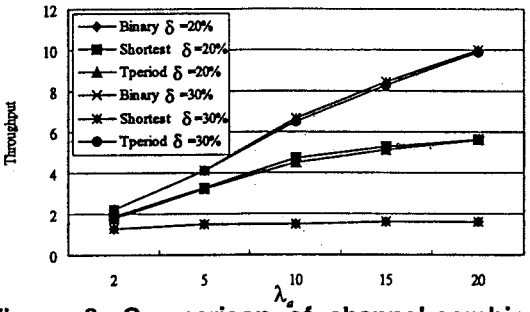

Figure 8. Comparison of channel-combine strategies on channel throughput at various arrival rates and slowdown ratios.

handoff rate $\lambda_h = 0.01$. In general, the SDF strategy is the best, the Partition strategy is only slightly worse than the former, and the Binary strategy is the worst.

B) Effect of arrival rate: In Fig. 9~11(a), we show the channel throughput offered by our three strategies when $\lambda_h = 0.01$. In general, when the λ_a increases, the throughput increases. In all observed scenarios, the channel throughput is larger than 1. One can regard throughput as the average number of mobile hosts on a channel at any moment. This demonstrates the advantage of using channel combination — channels do have high probabilities to merge together.

In Fig. 9~11(b), we show the corresponding dropping probability incurred by these strategies. The knees in the figures (where the dropping probability increases sharply) indicate how much arrival rate the strategies can sustain without becoming saturated. As can be seen, the knees range from $\lambda_a = 1$ to 10. For instance, when $\lambda_a = 4$, as the service time per user should range from 10 to $10/(1-\delta)$, by Little's Law, an average of 40 to $40/(1-\delta)$ users can be supported concurrently by a cell with only $n = 20$ channels.

C) Effect of Slowdown Ratio: The effect of slowdown ratio can also be observed from Fig. 9~11. The results show that larger δ will favor our strategies as there is more chance for channels to be combined.

5 Conclusion

We are currently investigating schemes to further improve the channel throughput. We are also investigating techniques to avoid the disturbance from handoff and new mobile hosts to our channel-combine strategies. The Partition strategy seems to be a good direction to the disturbance problem, although it is worse than the SDF strategy in the current stage. Finally, a more challenging problem is to consider the case that each mobile host has a different allowed slowdown ratio.

References

[1] A. Archarys and B. R. Badrinath. A framework for delivering multicast messages in networks with mobile hosts. *ACM/Baltzer J. of Mobile Networks and Applications*, 1(2):199–219, 1996.

[2] B. Badrinath, A. Acharya, and T. Imielinski. Impact of mobility on distributed computations. *ACM Operating Systems Review*, 27(2):15–20, 1993.

[3] K. Brown and S. Singh. The problem of multicast in mobile networks. In *5th International Conference on Computer Communications and Networks*, Oct. 1996.

[4] J. Geier. *Wireless Networking Handbook.* New Riders Publishing, Indianapolis, USA, 1996.

[5] A. Hills and D. B. Johnson. Wireless data network infrastructure at Carnegie Mellon University. *IEEE Personal Communications*, 3(1), Feb. 1996.

[6] L.-S. Juhn and L.-M. Tseng. Harmonic broadcasting for video-on-demand service. *IEEE Trans. on Broadcasting*, 43(3):268–271, Sept. 1997.

[7] L.-S. Juhn and L.-M. Tseng. Staircase data broadcasting and receiving scheme for hot video service. *IEEE Trans. on Consumer Electroincs*, 43(4):1110–1117, Nov. 1997.

[8] L. Kleinrock. Nomadic computing - an opportunity. *ACM Computer Comm. Review*, pages 36–40.

[9] L. Kleinrock. *Queuing Systems.* Wiley-Interscience, 1975.

[10] C.-J. Su and L. Tassiulas. Broadcast scheduling for information distribution. In *IEEE INFOCOM*, ?? 1997.

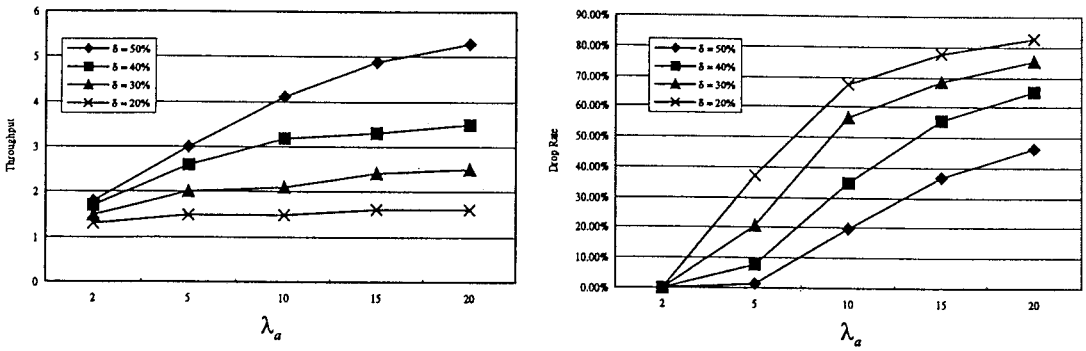

Figure 9. Performance of the Binary strategy: (a) throughput and (b) drop probability.

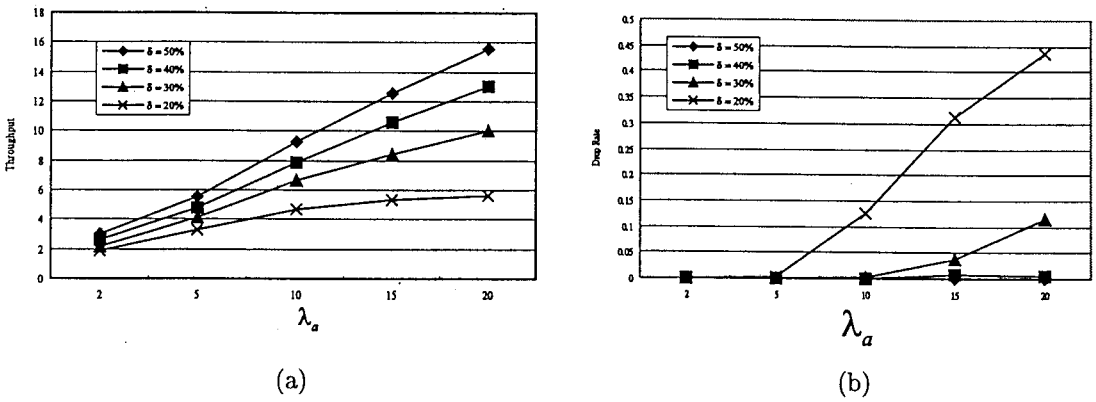

Figure 10. Performance of the Shortest-Distance-First strategy: (a) throughput and (b) drop probability.

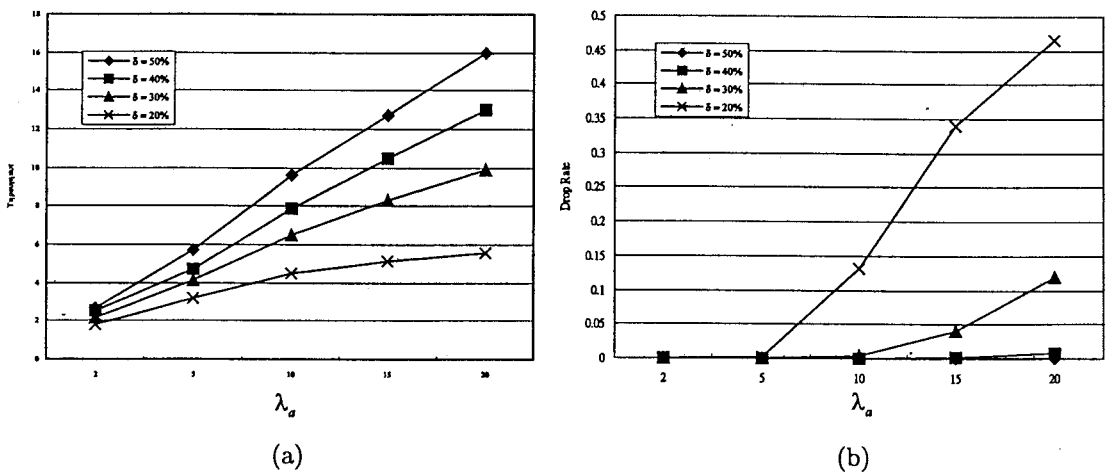

Figure 11. Performance of the Partition strategy: (a) throughput and (b) drop probability.