Distributed Data Management
Part 2 - Data Broadcasting in Mobile Networks
Today’s Questions

1. Data Access in Mobile Environments
2. Scheduling Broadcast Disks
3. Client Caching
4. Indexing Broadcast Disks
With the availability of mobile communication networks we have to re-consider the problem of data access to servers taking into account the changed physical environment. A typical situation is having a base station that covers a wireless cell, in which many mobile clients communicate with the base station. The base station provides access to information systems using a fixed network infrastructure. The characteristics of the environment can be fairly different depending on the underlying networking technology (e.g. GSM phones, WLAN or satellite). The problem we will consider in the following is where clients want to gain access to data that is accessible through the base station.

With respect to the network the following characteristics make the problem of data access different from the access through conventional, fixed networks.

1. The network is (or better can be) asymmetric, such that the bandwidth for sending data from the server to (all) clients (downstream) in the cell is much higher than vice versa (upstream). On the one extreme it is possible that clients have no possibility to send data to the server (e.g. a satellite), on the other extreme they bandwidths can be the same (e.g. a WLAN). Still, in the latter case all mobile units would compete for the same bandwidth.

2. A second important issue is power consumption. Energy is a scarce resource at the mobile unit, and often active communication is by orders of magnitude more energy-consuming than switching to a doze mode. So techniques for minimizing the active time of a mobile unit are important.

3. Frequent disconnections can occur, e.g. when mobile units change a wireless cell. This poses problems in terms of data consistency, e.g. when at that moment an update transaction is executed and is aborted. Basic assumptions on transactions in fixed networks, e.g. having a reliable communication channel, no longer hold. We will not explore this third issue here in the following.

Another typical characteristics in mobile environments is that often the number of clients is specifically high as compared to the number of servers.
Exploiting Asymmetry: Data Broadcast

- High downstream bandwidth $B$ as compared to upstream bandwidth $U$
  - e.g. satellite downlink with 400Kb/s
  - sometimes no upstream connection available

- High number of clients $C$ as compared to number of requests $R$
  - e.g. stock quotes, tickers, newsletters etc.
  - therefore better to share downstream capacity among requests: $B/R > B/C$

- If $B$ sufficiently large then $B/R > U$
  - more efficient to simply immediately send all answers to potential requests over the downstream broadcast channel than to wait for individual requests of clients: data broadcast or data push
  - capacity independent of number of clients $C$

Asymmetry in a communication environment means that the down-stream capacity is substantially larger than the upstream capacity. The capacity is however not only related to the bandwidth of the communication link, which can be asymmetric in a mobile environment, but also to the distribution of data that has to be transmitted over the communication link. In general, with a growing number of clients it is likely that the number of different data items that need to be transmitted over the communication link grows slower than the client population. For example, in a news ticker scenario it is very well possible that the number of news items is much lower than the number of clients. This situation in fact can occur also in fixed networks, and therefore some of the techniques we will be looking at in the following apply there as well.

At this point we can make two observations:

(1) if the number of clients $C$ increases, the number of different requests $R$ may increase slower. Thus the downstream capacity per request will be higher than the downstream capacity per client. In other words if we can share the downstream capacity for multiple clients when they have the same request the downstream capacity is higher in total.

(2) if $D$ is reasonably large, $R$ is not too large and the upstream capacity $U$ is low, then it can occur the downstream capacity per request becomes large than the upstream capacity. If we assume that a request and response in a client-server interaction consume approximately the same bandwidth (which is acceptable for data requests), this implies it makes no more sense to issue requests from clients, but the server simply sends all possible answers over a broadcast channel. Since $D/R$ is independent of the number of clients $C$ this approach scales arbitrarily well in the number of clients. If $U=0$, i.e. clients have no upstream capability, broadcast is the only possible approach in any case.
Periodic broadcast is a simple way to push data to the clients. The server selects the data items to be disseminated, defines an order on them and sends them in this order periodically over the data channel. We assume that the data items of the same size and that they are identified by a key (in a relational database these could be fragments of relations, but as well pages of the storage system, depending on the level of abstraction that is used to provide the data to the client, in an XML database these could be XML documents). When the client receives this stream of data items it appears as if the whole database is stored on a disk with a peculiar access characteristics. In particular, the latency of this access can be very high as the "broadcast disk" does only support sequential access to the data.

An important problem in access to the broadcast is whether the client has to actively listen. Many mobile devices distinguish a doze mode from an active mode, which consumes by orders of magnitude less energy. In case the client has additional information on the time when interesting data items will occur on the broadcast disk, it has the possibility to switch to the doze mode in those times where no relevant data occurs.

In the following we will make some additional assumptions on the characteristics of the problem. In particular, we will assume that the problem is static, and that clients do not actively communicate with the server, in particular they cannot make updates to the database. Hybrid approaches where communication can occur in both directions have been investigated but are beyond the scope of this lecture.
A very popular example of a broadcast medium is videotext, which exhibits many interesting characteristics of the problem. Actually we can observe also there the problem of "tuning time", although not related to power consumption, but related to the time we have to stay at a specific TV channel to obtain the page (assume that you are not interested in the current TV program). The numbers are somewhat arbitrary and just serve for illustration.
Summary

• How does a mobile environment differ from a fixed network, when accessing databases?

• When does it make sense to broadcast data?

• What are the criteria for optimizing access to a data broadcast?
What Do You Think?

• How can we optimized latency and tuning time for periodic data broadcast (e.g. videotext)?
2. Scheduling Broadcast Disks

- Naïve Approach: Flat organization
  - Set of all data objects is cyclically broadcasted
  - Expected delay = N/2, N number of data items

- Key Idea for improvement: Take into account the access frequencies to data objects when scheduling a broadcast
  - More frequently used items should appear also more frequently on the broadcast disk

The naïve approach to creating a data broadcast is to organize it in a flat fashion, i.e. each data item occurs exactly once. Then it is easy to see that if the broadcast contains N data items that the expected time to wait for a specific data item starting at some random time is N/2. In practice different data items are accessed with different frequency and therefore a possible optimization would be to include more frequently requested data items more often into the broadcast. In the following we introduce a method for doing this assuming that access frequencies are known.

Remark: you might have noticed that the Videotext system is doing something similar: main pages appear in general much faster than less popular ones.
Example

- Assume A is accessed more frequently: \( p_A > p_B, p_C \)

\[
\begin{array}{c}
\text{(a) flat broadcast disk} \\
\text{(b) skewed broadcast disk} \\
\text{(c) multilevel broadcast disk}
\end{array}
\]

Observations
- (b) is always worse than (c) in terms of average delay
- Expected delay depends on the ordering of data items on the broadcast disk
- In (b) and (c) access to B and C becomes slower
- A is on a broadcast disk which "spins twice as fast" as the disk on which B and C are stored: multilevel broadcast disk

<table>
<thead>
<tr>
<th>Access Probability</th>
<th>Expected Delay (in broadcast units)</th>
</tr>
</thead>
<tbody>
<tr>
<td>( p_A ) ( p_B ) ( p_C )</td>
<td>Flat</td>
</tr>
<tr>
<td>---------------------</td>
<td>------</td>
</tr>
<tr>
<td>0.33 0.33 0.33</td>
<td>1.50</td>
</tr>
<tr>
<td>0.50 0.25 0.25</td>
<td>1.50</td>
</tr>
<tr>
<td>0.75 0.125 0.125</td>
<td>1.50</td>
</tr>
<tr>
<td>0.90 0.05 0.05</td>
<td>1.50</td>
</tr>
<tr>
<td>1.00 0.00 0.00</td>
<td>1.50</td>
</tr>
</tbody>
</table>

This example illustrates that when replicating more popular data items in the data broadcast, the interleaving of data items can be organized in different ways. Assume that A is more popular than B and C and therefore it has been decided to include two copies of A into the broadcast. There exist two possibilities for doing this: either sending two A's consecutively (which results in a skewed broadcast) or to interleave A with the two other data items. The interleaved broadcast is also called a "multidisk" broadcast, since from the viewpoint of the client it appears, as if the broadcast consists of two different disks it can alternatively access, but the speed at which the "disk" containing A rotates is twice the speed at which the "disk" containing B and C rotates.

The table gives the calculated values for the expected delays for different combinations of access frequencies. Of how these are precisely calculated is shown on the next slide, but qualitatively it is no problem to understand the result. Some of the observations are:

- for the flat organization the expected delay (averaged over all data items !) does not change.
- the skewed organization is always worse than the multidisk configuration. So we see that the scheduling of data items in the data broadcast is not only a bandwidth allocation problem (then skewed and multidisk would be the same), but has to take into account ordering.
- The "break-even" point where the multidisk configuration starts to pay off is reached when A is accessed more than half of the time

What cannot be seen from the table is that in fact the access to B and C become slower in the multidisk configuration, and therefore for a uniform distribution of access frequencies the multidisk configuration is worse. This loss is however compensated increasingly by the fact that the more frequently requested data item A is accessed faster.
Computation of Expected Delay

The computation of the expected delay is not very complicated but requires some care. For a given data item $d$ the delay is computed for each interval of the data broadcast separately. One assumes that if the request arrives within the interval $i$ where the requested data item $d$ lies, then one has to wait for the next transmission of this data item, as the complete interval is required to obtain the complete data item.

For example, for broadcast scheme (b) (skewed broadcast) the expected delay for a request for $A$ is computed as follows:

Arrival of request in interval 1: $t_{i,d}^{d_{\text{max}}} = 1$ (next interval), expected delay 0.5
Interval 2: $t_{i,d}^{d_{\text{max}}} = 3$ (beginning of next broadcast), expected delay 2.5
Interval 3: $t_{i,d}^{d_{\text{max}}} = 2$ (beginning of next broadcast), expected delay 1.5
Interval 4: $t_{i,d}^{d_{\text{max}}} = 1$ (beginning of next broadcast), expected delay 0.5
Total: $(0.5+2.5+1.5+0.5)/4=1.25$

For B and C we have delay = 2

Therefore for equally distributed probabilities, we obtain an expected value of $1/3*1.25+1/3*2+1/3*2=7/4=1.75$
Theorem 1: The broadcast schedule with minimum overall mean access time results when the instances of each data item are equally spaced.

Theorem 2 (Square-Root Rule): Assuming that instances of each data item are equally spaced, minimum overall mean access time is achieved when the frequency \( f_i \) of each data item \( d_i \) is

\[
f_i \propto \sqrt{p_i}
\]

The overall mean access time is then

\[
delay_{optimal} = \frac{1}{2} \left( \sum_{i \in D} \sqrt{p_i} \right)^2
\]

Remark: Since equal spacing cannot necessarily be always obtained this gives a lower bound

For finding good broadcast disk organizations we will make use of two theoretical results.

The first generalizes our observation on skewed broadcasts, which are always worse than multidisks. It can be shown that the mean access time to a broadcast is minimized if ALL data items are distributed over the broadcast such that the space between them is constant. So any broadcast organization should try to achieve this property. One has to be observe however, that for concrete cases this property not necessarily can be achieved. For example, if we would have to distributed 2 data items A and 3 data items B, there exists no broadcast that allows equal spacing. So only approximations of schedules with equal spacing will be possible in general.

The second result applies in case data items are equally spaced. It says that then it is possible to determine the optimal frequencies with which data items have to occur in the data broadcast, provided that the access probabilities are known. The interesting observation is, that these two quantities are not proportional (as one might expect at the first moment !), but rather that the frequency in the broadcast has to be proportional to the root of the access probability. This means that the popular data items appear in general less frequently in relation to the number of accesses that are made to them.
Broadcast Scheduling Algorithm

• Find scheduling algorithm that creates
  - periodic broadcast
  - with fixed inter-arrival times for data items
  - optimized frequencies of data items
  - and uses under these constraints as much bandwidth as possible

• Algorithm Overview
  1. order data items according to frequencies and cluster data items with same (similar) frequencies: defines broadcast disks
  2. determine the optimal frequencies for the broadcast disks
  3. distribute the data items evenly spaced over the broadcast disk

Given the two results mentioned before the task is now to find a broadcast organization that satisfies the conditions on an optimal broadcast as closely as possible (we have seen already that we ill in general not be able to satisfy them exactly).

The general approach will consist of three main steps: first identify those data items that should occur with the same frequency in the broadcast, since they are accessed with the same (or similar) probabilities. This defines the different "disks" of the broadcast. Then based on the access probabilities to each of the disks optimal frequencies of occurrence are computed according to Theorem 2. In the last step a broadcast disk is configured such that the data items of each disk occur exactly with the (approximate) optimal frequency determined and that the data items are equally spaced.
Determining Optimal Frequencies

- Broadcast disks $D_1, ..., D_k$ with access probabilities to each data item of $p_1, ..., p_k$ have been defined, such that $p_1 > p_2 > ... > p_k$
- Then ($f_{\text{min}}$ is freely chosen)
  \[ f_j = f_{\text{min}} \left\lfloor \frac{p_j}{p_k} \right\rfloor, \ j = 1, ..., k \]
- Example:
  - $d_1$ with access probability $1/2$
  - $d_2$ and $d_3$ with access probability $1/8$
  - $d_4 - d_{11}$ with access probability $1/32$
  - 3 broadcast disks $D_1=\{d_1\}, D_2=\{d_2,d_3\}, D_3=\{d_4,d_5,d_6,d_7,d_8,d_9,d_{10},d_{11}\}$
- Optimal frequencies ($f_{\text{min}}=1$)
  \[ f_1 = \left\lfloor \frac{\frac{1}{2}}{\frac{1}{32}} \right\rfloor = 4 \quad f_2 = \left\lfloor \frac{\frac{1}{8}}{\frac{1}{32}} \right\rfloor = 2 \quad f_3 = \left\lfloor \right\rfloor \]

Defining the different disks and determining their corresponding access probabilities is trivial provided we have a set of data items with access probabilities given. This first step is illustrated in the example. The next step is to determine the optimal frequencies. Since the theorem only determines the proportion among the access probabilities and frequency of data items in the broadcast, we can choose one of the frequencies freely. For practical reasons, since we want to obtain integer frequency values we choose the frequency of the data item with smallest probability of access freely ($f_{\text{min}}$). From there all the other frequencies are determined due to Theorem 2. We have to multiply the minimal frequency with the square root of the proportion of the smallest access probability and the access probability for the disk for which we determine the frequency. Since the frequency must be an integer (remember: the frequency is the number of copies of the data item/disk in the broadcast), we take the next lower integer value. A sample computation of frequencies is given for our running example.
Broadcast Schedule Generation

• Problem: how to distribute data items evenly spaced with correct frequencies?

• Example: d1,d1,d1,d1,d2,d2,d3,d3,d4,d5,d6,d7,d8,d9,d10,d11

• Idea:
  distribute $D_1$ ($f_1=4$)
  $d_1$ $d_1$ $d_1$ $d_1$
  interleave $D_2$ ($f_2=2$)
  $d_1$ $d_2$ $d_1$ $d_3$ $d_1$ $d_2$ $d_1$ $d_3$
  interleave $D_3$ ($f_3=1$)
  $d_1$ $d_2$ $d_4$ $d_5$ $d_1$ $d_3$ $d_6$ $d_7$ $d_1$ $d_2$ $d_8$ $d_9$ $d_1$ $d_3$ $d_10$ $d_11$

• This works since $\text{size}(D_3) \times f_3 = 8$ and $\text{size}(D_2) \times f_2 = 4$ are multiples of $\text{size}(D_1) \times f_1 = 4$.

• What if this is not the case?

Now that the frequencies are known for each disk, we have to distribute the different disks according to the frequencies over the broadcast. This is in fact not a completely trivial problem. In our example we could start as indicated above by first distributing disk $D_1$, then interleaving the data items from disk $D_2$, by alternating the occurring multiple items, and then interleaving the data items from disk $D_3$. The resulting broadcast satisfies all required properties, e.g. the data item from disk $D_1$ appears 4 times (frequency 4) and all data items are equally spaced. However, this works only due to the specific structure of this example.
Broadcast Schedule Generation

- Partition each disk $D_i$ into a number of smaller units (chunks) $C_{ij}$, $j=1,...,c_i$
  
  $$c_i = \frac{c_{max}}{f_i} \quad \text{and} \quad c_{max} = \text{LCM}(f_1,...,f_k) \quad \text{(least common multiple)}$$

- Then generate $c_{max}$ copies of each chunk $C_{ij}$ and distribute them evenly

- Consequence: for disk $D_i$: $\frac{\text{size}(D_i)}{c_i} \times c_{max} = \text{size}(D_i) \times f_i$

- Therefore each data item in $D_i$ appears exactly $f_i$ times in the broadcast!

- If $\text{size}(D_j)$ cannot be divided by $c_i$, fill it up with data items to disk $D_j^*$ till $\text{size}(D_j^*)$ can be divided by $c_i$
  - The data for filling up can be frequently used data or other information such as indexes and updates
  - However, the number of disks will be small as compared to the number of pages, therefore few gaps occur in practice

In case the frequencies are not in such a "nice relationship" as before (e.g. assume $f_1=2$, $f_2=3$, $f_3=5$), we have to apply a "trick". The idea is to find partitions of the set of data items in each disks into chunks in a way that when later producing a constant number $c_{max}$ of copies of each of the chunks all data items occur with the expected frequencies. Since the number of copies of chunks is constant it is then straightforward to generate a broadcast schedule by simply regularly distributing the $c_{max}$ copies of the chunks over $c_{max}$ bins.

In order to properly partition the disks into chunks we have to determine depending on the required frequency the number of chunks. This can be done in the described above, starting out from the least common multiple of all frequencies. The computation above also shows that by choosing the numbers of chunks in this way generates the desired frequencies for each disk.

Of course not every disk can be precisely divided into the desired number of chunks. For example, taking the frequencies $f_1=2$, $f_2=3$, $f_3=5$, $c_1$ would be 15. Now if disk $D_1$ contains a single data item only it can obviously not be divided into 15 pieces. Therefore, what is done, is to fill the disk up with other information (in this case 14 more data items) to make it divisible. In practice the problem is much less severe as it appears in this extreme case, since normally the number of data items on each disk will be large as compared to the frequencies values involved and thus only very few data items need to be added.
Once the chunks are determined it is fairly straightforward to distribute them over the $c_{\text{max}}$ bins. A possible algorithm doing that is given above. Note that by distributing the chunks regularly over $c_{\text{max}}$ bins, the broadcast schedule consists of two types of cycles: a minor cycle, were periodically each disk is having some (possibly varying) data items, and a major cycle which is the whole broadcast schedule.
Summary

• Why is a flat broadcast disk organization not necessarily optimal?

• What is the effect of ordering multiple data items in a broadcast disk on latency?

• What is the square-root rule for broadcast disk organizations?

• How are the frequencies of the major and minor cycle in a broadcast disk computed?