Mobile Cache Management:

Cache consistency problem is to ensure that the cached copy at a client is up to date w.r.t. the master copy at the server.
- Lack of global clock.
- Failures complicates implementation.
Two types of Consistency Maintenance Scheme

1. Stateful (push) - Server is responsible for ensuring consistency of client caches.
2. Stateless (pull) - Client is responsible - so whenever it accesses a cached copy it makes sure that the copy is up to date.

- Maintains which clients are caching a particular data item.
- Whenever a data item changes it informs all the clients which are caching that particular data item to invalidate their cached copy.

Problem: Invalidation message can get lost.

1. Frequent updates cause flooding problem.
2. Maintaining state information can be a huge burden on the server → may affect the scalability.
1. Modification timestamp to verify if a client has up-to-date copy

2. Write through scheme which ensures that server is aware if I am a client-modifying its cached copy.

Which of the above two approaches is better for MCE?

- For (non-mutable) data, stateful is better?

- Resource Consumption? Bul > BDL  
  - B/U (wireless) — > downlink BDL
  - Energy of Client: E_{Tx} > > E_{Rx}
To cache or not to cache?  Sec. 3.4.6.

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ws</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Fs</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

windowing algorithm for data allocation at mobile client.

- Dynamic online algorithm
- Maintain a window of size \( K \) of last \( K \) "relevant" operations. \( (W_s, R_c) \)
- If \( W_s \) or \( R_c \) are greater than \( K/n \), then don't cache otherwise cache.
What is the impact of window size $k$?

"$k$" determines how agile the algorithm is in adapting.

How to implement the windowing protocol?

Broadcasting Timestamp