Lecture 10

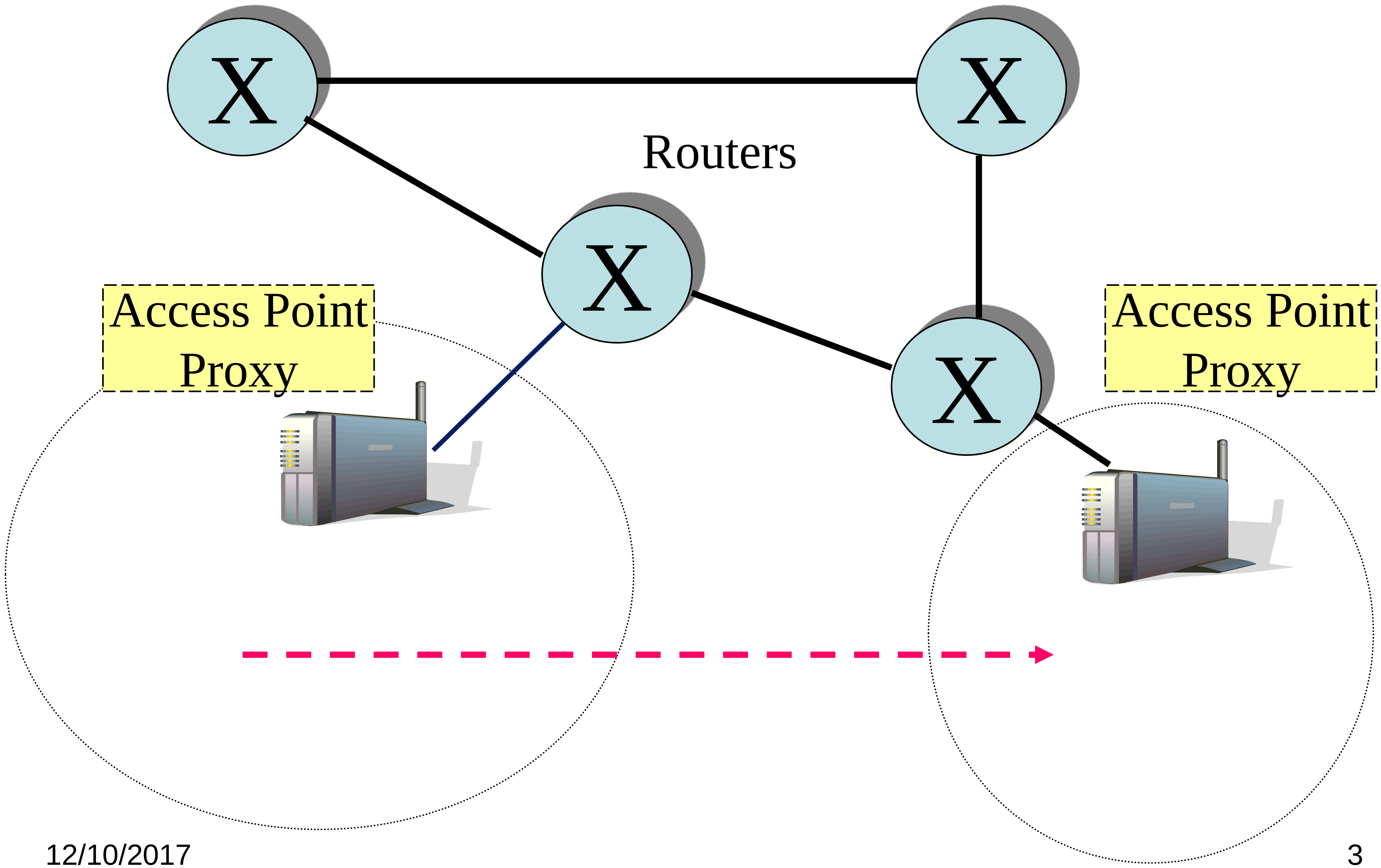
Mobility management service

The context: AP based mobile networks

- The problem:
  - while on the move, mobile devices subscribe to events or need to fetch data from Internet servers (e.g., for synchronization purposes). Disconnecting/connecting to Access Points (AP) should be *transparent*.
  - The middleware has to provide a mobility and data consistency support service that will minimize duplication and loss of information.
  - This service can be implemented by *proxies* and *caching*.
  - As caching raises the problem of consistency, one solution is to have the AP periodically broadcast *invalidation reports (IR)* that contain data that was changed during the last quantum.

- Supporting architecture:
  - APs are part of a distributed system, mobile clients – Internet servers.
  - A proxy is a stationary component running at each AP or close to it.
  - While the client is disconnected and during the hand-over phase, subscriptions and publications are managed by the proxy/caching system.

The System Model



A. Mobile publish/subscribe service

- Mobile clients use a *mobility service client library*, linked to the client application.

- The general protocol:
  1. When the client is connected, the client library mediates subscriptions – it stores a copy of client subscriptions.
  2. Before disconnecting, the client calls the *move-out* function which causes the client library move the stored subscriptions to the mobility proxy, on the AP.
  3. The proxy proceeds to subscribe and buffer all incoming messages.
  4. When the client reaches the destination, it calls the *move-in* function, to contact the local proxy.
  5. The two proxies execute a protocol that results in the transfer of all subscriptions and buffered messages.

Move-out

- When the client invokes the move-out, the client library passes to the move-out proxy the following:
    1. a unique client identifier;
    2. a list of subscriptions;
    3. an optional QoS specification.

- The *mobility proxy* executes the move-out function by creating a *client handler* with the client id and subscribes it according to the subscription list.
- When the proxy is finished, it’ll send an ack to the client.
- The client library disconnects the client from the publish/subscribe system and returns from the move-out function.

Move-in

- By calling this function, the client either reconnects to the same proxy or to the move-in proxy.
- In the former case, the client library activates a queue and starts up a receiver process; then, restores all client' subscriptions.
- In the latter, the client library sends a download request to the move-in proxy, giving the address of the move-out proxy and the client id.
- The two proxies get in contact and download all messages. The proxy forwards all messages to the client library.
- The move-out proxy un-subscribes its client handler and destroys the handler.

Synchronization options

- Messages may be duplicated:
  - during the move-out function;
  - during the move-in function.
- Messages may be lost:
  - during the move-out function;
  - during the move-in function.

- One mechanism is to use ping messages between the two proxies, using the P/S system itself. Another possibility is to introduce a delay before removing subscriptions issued by move-out proxy.

B. Data caching approaches

- Existing cache invalidation algorithms can be classified into,
  - the *stateful server* which maintains state information for each client:
    - the asynchronous and stateful cache invalidation (AS);
  - the *stateless server* doesn’t have any state information of the clients:
    - the invalidation report algorithm (IR); it is extended with updated IR.
- The invalidation report (IR) is broadcasted by the server every L seconds. Its purpose is to announce a list of recently updated data. Each client compares its cache with the list in IR and isolates any obsolete ones. The remaining copies are valid and suitable for serving future queries.

The invalidation report

- Formally, each IR is a set of tuples:

$$IR_i = \{(d_x, t_x) \mid (T_i - wL) < t_x \leq T_i\}$$

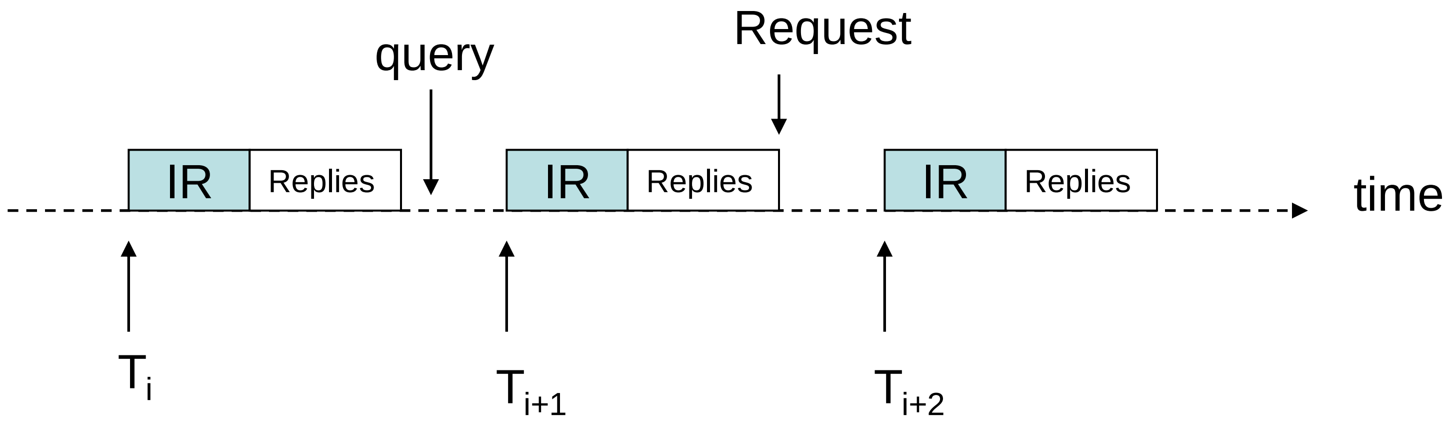
- Where d_x is the index of a data item, and t_x is the timestamp of its most recent update. T_i is the most recent timestamp and w is the broadcast window size, which controls the amount on invalidation information to be included in each IR.
- Clients do not use directly their caches to answer queries. Instead a client waits for the next IR to determine whether its cache is valid or not. If a valid cached copy exist, the query can be served locally. Otherwise, the client issues an uplink request to the server and waits for the reply after the next IR.
- If a client is disconnected from a server for a period less than wL, it can still use the IR to invalidate any obsolete items. Thus, another function of w and L is to control the longest disconnection time that can be supported by the algorithm.

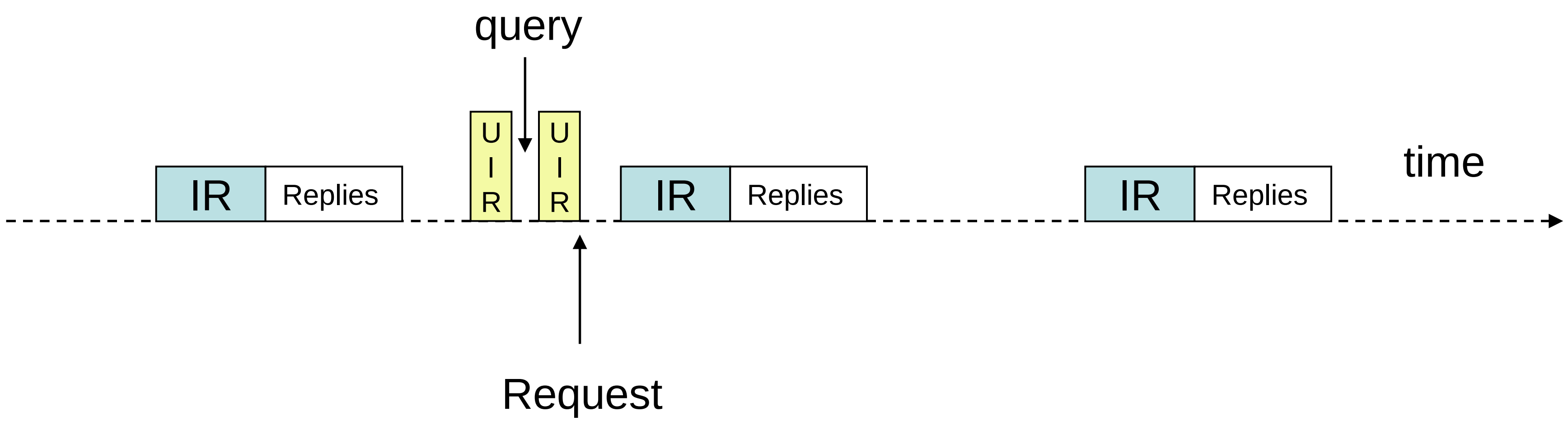
The Updated Invalidation Report

- As IRs are scheduled to be broadcasted periodically, the clients can save battery power by listening around the broadcast time. There are benefits and drawbacks…
- To reduce the long query delay associated with the basic IR approach, an updated invalidation report (UIR) was proposed.

- Each UIR contains invalidation information for those data that have been updated since the last IR.

- Clients use UIR to invalidate their caches. If there is a hit, the query can be answered earlier. Otherwise, clients can also send an uplink request at an earlier time. In both cases, the query delay can be reduced.


query
Request
IR
Replies
IR
Replies
IR
Replies
time
T_i
T_{i+1}
T_{i+2}


query
IR
Replies
U I R
U I R
IR
Replies
IR
Replies
time
Request

Asynchronous and Stateful algorithm

- Each mobile device is associated with a home location cache (HLC), which maintains a list of caches kept by the mobile host – the state information.
- When the mobile device connects to a foreign network, HLC is duplicated at the AP, in order to ensure that an HLC is always available for the mobile device.
- The state information is a set of tuples, (d_x, t_x, flag), where d is the id of the data element and t is the timestamp of its last invalidation. The flag is true when an invalidation has been sent to the mobile device but no ack was received.
- The distinguishing feature of this algorithm is that the cache invalidation information is sent asynchronously and also buffered at AP until an explicit ack is received.
- The validity of the mobile's cache should be ensured before answering any query. The AP helps with that: a data server informs the AP when a data item is updated; the AP determines from HLCs the set of mobiles that cache that data. Then, the AP sends the invalidation information by uni-casting. The mobile invalidates its cache copy.

Supporting disconnection

- Each client keeps a cache timestamp which corresponds to the last received message. The client includes the cache timestamp in every message to the AP.
- When a mobile device reconnects, it sends a probe message with its cache timestamp, t.

- The AP consults the corresponding HLC to:
  - discard all invalidation records with timestamp less than or equal to t;
  - send all invalidation records with timestamp greater than t.

- The mobile defers all queries until it receives the ack for its probe message. This way, it determines the set of data being updated during its disconnection.

Application

- When moving from one AP to another one, the mobile goes through a time consuming authentication process to perform a key exchange with the new AP.

- Proactive key caching (PKC): the 1st time a mobile connects to an AP it receives a key during the authentication process – this is cached. Then it might move, disconnect, connect to other APs and, later, reconnect to the first AP. In that case it will re-use the cached key. This can reduce the authentication time with up to 60%.