

## Motivation

- Data access issues in ad hoc networks:
  - Long query delay due to multi-hop links
  - Low data availability due to node/link failure
- Cooperative caching schemes
  - Share and coordinate cached data.
  - Reduce query delay, bandwidth/power consumption.

- Battle field:
  - Officer may have a powerful data center. Soldiers need to access the data center to get geographic info, enemy info, and new commands. Neighboring soldiers share these info.
- InfoStations
  - Infostations are deployed in cities to provide info, such as maps, attractive sites, or restaurant info to mobile users. Users may relay for each other to serve those not directly covered by the infostations.

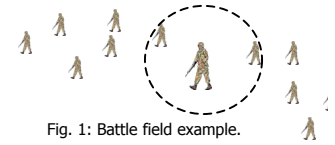


Fig. 1: Battle field example.

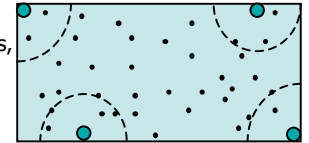


Fig. 2: InfoStation Example

Green node: infostation; Black node: mobile node

## Data Access Framework

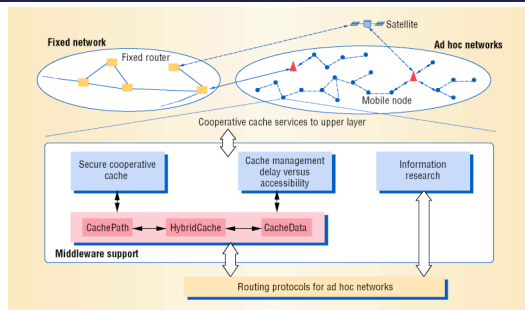


Fig. 3: Cooperative cache-based data access framework

- The cooperative cache-based data access framework stays on top of the routing protocol.
- It relies on several components such as
  - Secure cooperative caching,
  - Cache Management,
  - Information search
 to provide services to upper layer.

## System Model

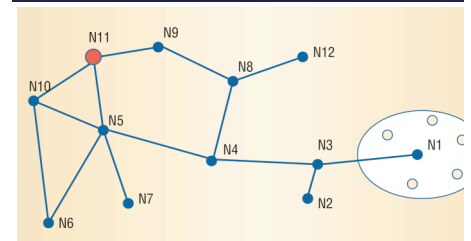


Fig. 4: Ad hoc networks. Node  $N_{11}$  is a data center and the blue are router nodes. Node  $N_1$  is a cluster header surrounded by cluster member nodes.

## The Cache Protocols

### The CacheData Scheme

- Router nodes caches frequently accessed passing-by data to serve future requests.
  - An example (using Fig. 4)
    - Suppose both  $N_6$  and  $N_7$  requested data item  $d_i$  through  $N_5$ .
    - $N_5$  might think that  $d_i$  is popular and cache it locally.
    - $N_5$  can then serve  $N_4$ 's request directly.

### The CachePath Scheme

- Cache the node  $id$  that requests the data when a data item passes by.
  - An example (using Fig. 4)
    - Suppose  $N_1$  requests  $d_i$  from  $N_{11}$ .
    - When  $N_3$  forwards  $d_i$  to  $N_{11}$ , it caches the id pair of  $N_1$  and  $d_i$ :  $(i, 1)$ .
    - Future requests for  $d_i$  from  $N_2$  can be redirected to  $N_1$  to reduce query delay and resource consumption.

### HybridCache- a Hybrid Scheme

```

(A) When a data item  $d_i$  arrives:
    if  $d_i$  is the requested data by the current node
    then cache  $d_i$ 
    else /* Data passing by */
    if there is a copy of  $d_i$  in the cache
    then update the cached copy if necessary
    else if  $s_i < \zeta_i$  or  $TTL_i < \delta_i$ , then
    cache data item  $d_i$  and the path;
    else if there is a cached path for  $d_i$ , then
    cache data item  $d_i$ ;
    else
    update the path for  $d_i$ ;
    else
    cache the path of  $d_i$ ;

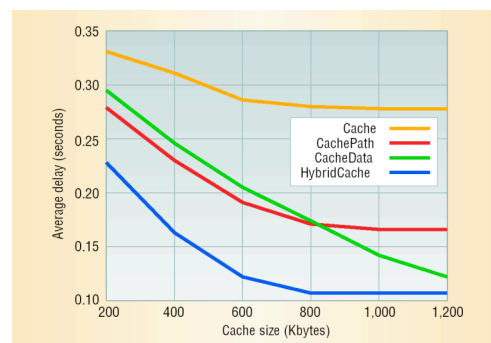
(B) When a request for data item  $d_i$  arrives:
    if there is a valid copy in the cache
    then send  $d_i$  to the requester;
    else if there is a valid path for  $d_i$  in the cache then
    forward the request to the caching node;
    else
    forward the request to the data center;
    
```

Fig. 5: HybridCache: take advantage of both CacheData and CachePath.

## Cache Management

- Cache replacement policies
  - To determine whether to replace  $d_i$ , node  $N_i$  considers:
    - $\delta$ : Distance of  $N_i$  to the data center or a caching node of  $d_i$ , small  $\delta$  value is preferred for removal.
    - $\zeta$ : last update time of  $\delta$ , obsolete data is preferred.
    - Combinations of  $\delta$  and  $\zeta$  can affect the tradeoffs between query delay and data accessibility.
- Cache admission control
  - Following similar rules used in cache replacement

## Experimental Result



## Conclusion

- A cooperative cache-based data access framework
  - Let mobile nodes cache the data or path to the data to reduce the query delay and improve data accessibility
  - Provide secure cooperative cache, cache management, and information search services to upper layer.