

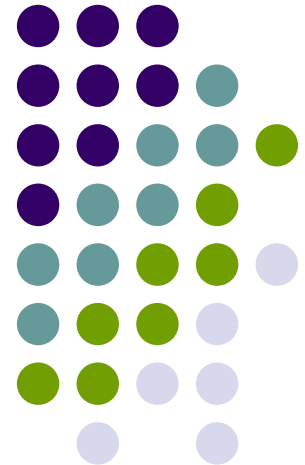
ANDROID

Android Opportunistic Networking

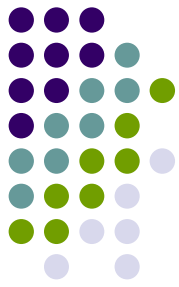


Ólafur Helgason

Laboratory for Communication Networks
School of Electrical Engineering
KTH - Royal Institute of Technology

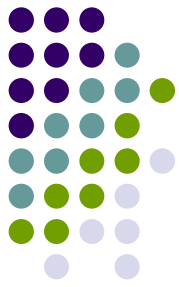


Outline



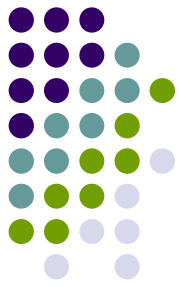
- Opportunistic Networking
- System overview
 - Opportunistic Content Distribution
- Implementation
- Evaluation
- Conclusions

What's ON?



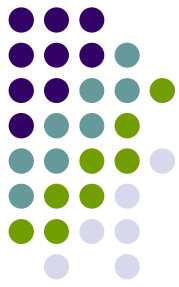
- Traditional communication mode
 - Phone-to-infrastructure
 - Cellular (GSM/3G/4G..)
 - WiFi Access Point
- Opportunistic Networking
 - Phone-to-phone
 - Pure P2P
 - Enabled by ad-hoc radio
 - 802.11 in ad-hoc mode
 - Bluetooth
 - No dependence on infrastructure

Why go ON?



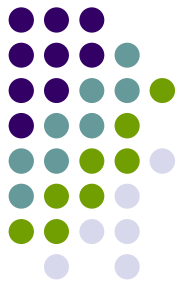
- Locality
 - Why use infrastructure if we don't have to?
 - Locality without positioning system (GPS or other)
- Scalability
 - “The more the merrier”
 - Not the case with infrastructure
- Offloading infrastructure
 - Immense growth of mobile data traffic
 - Ericsson, march 2010: “Mobile data traffic surpasses voice”
 - Forecasted to double annually next 5 years
 - Will infrastructure capacity keep up?
 - Capacity: Remains a bottleneck, upgrades expensive
 - Pricing: “Flat-rate” improbable if capacity limited
 - Alternative approaches for distribution important
 - Opportunistic, dynamic spectrum, etc.

Why go ON?



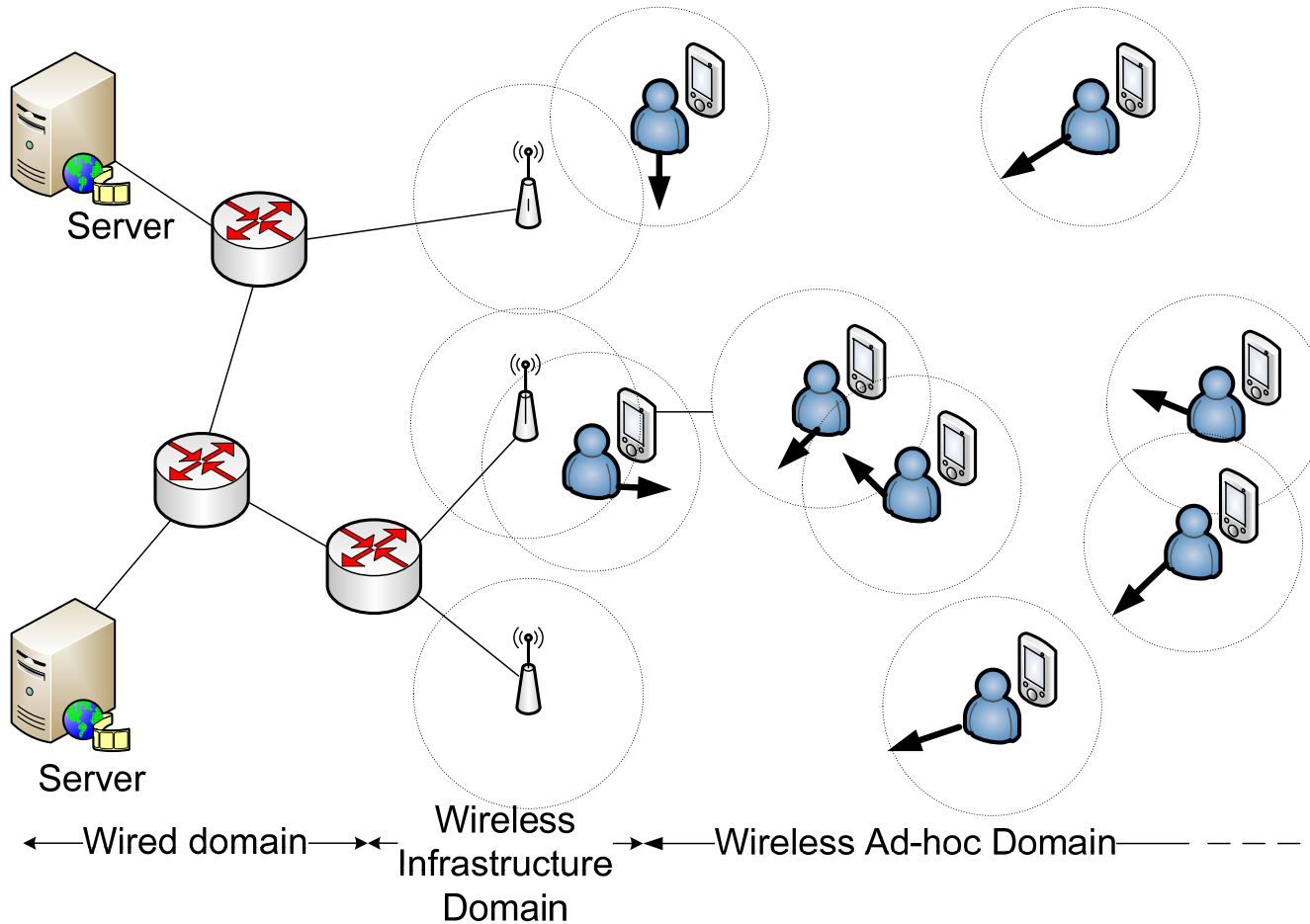
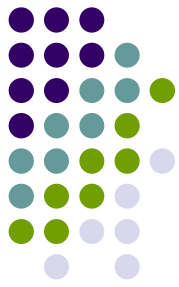
- Fault-tolerance
 - No single point of failure
 - Rescue, disaster & military applications
- What applications are suitable?
 - Must be tolerant to modest amount of delay
 - E-mail, www
 - Some location based services
 - Subscription services
 - Podcast, blogs, Facebook/twitter feeds

Opportunistic Content Distribution

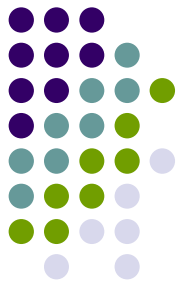


- Nodes share content with peers
- Content exchanged when nodes in range
- Content disseminates via sharing and mobility
 - Completely decentralized
 - No dependence on infrastructure
 - Free & open
 - Network neutrality, No “Big Brother”
- Content-centric applications
 - Blogs, media-sharing, voting, bulletin board, social, location, ...
- Goal of work
 - Specify a generic middleware architecture for Android
 - Facilitate simplicity in application development
 - Implement
 - Evaluate

System Overview

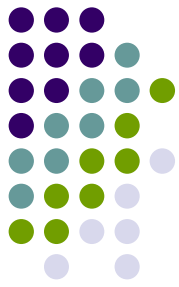


Content Structure

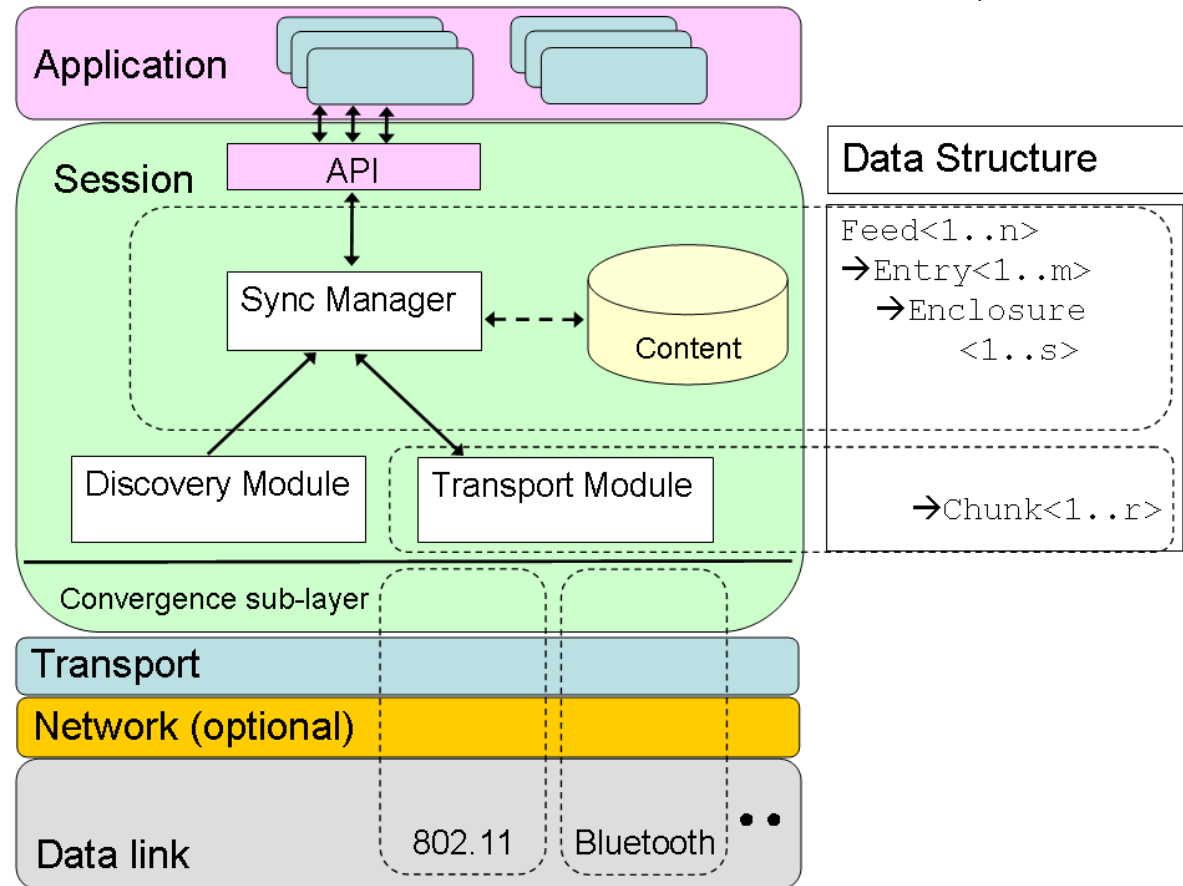


- Structuring content
 - Efficient unique matching of content queries
 - Logical grouping of topics
 - Facilitates content search
- **Feed:** Groups a logical topic
 - **Entry:** The objects of interest
 - **Enclosure:** The actual file (mp3, jpg, avi...)
 - **Chunks:** Download unit, fixed size
- Opportunistic publish/subscribe system
 - Entries are *published* on feeds
 - Nodes *subscribe* to feeds
 - *Notified* when new entry downloaded

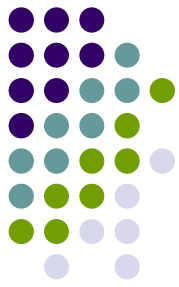
Node Design



- Middleware implements interest-based dissemination
 - Hides distribution, connectivity disruptions, etc.
- Major components
 - Application Programming Interface
 - Synchronization manager
 - Node/service discovery
 - Solicitation protocol

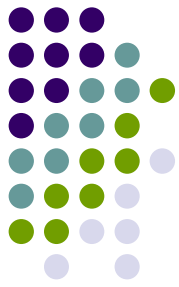


Implementation



- Implemented on Android Platform
 - Deployed on HTC Hero mobile phone
- Middleware implemented as an *Android Service*
 - Background process
 - One sample application
 - Opportunistic Photo/Video Blog
- Applications can *bind* to service
 - Wrapper for IPC mechanism
 - Connect to a different process
 - Services can export an API
 - Applications can register a callback function

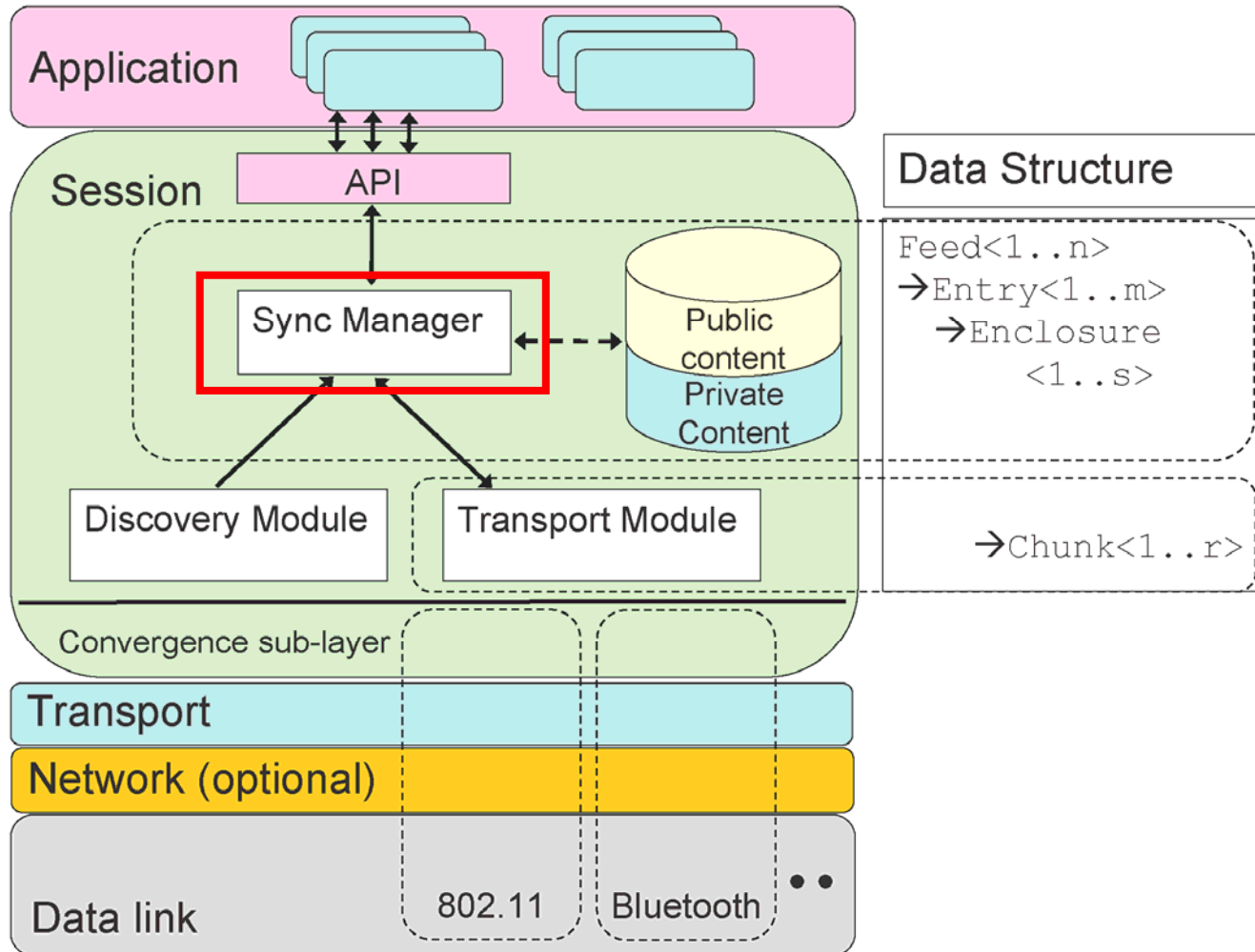
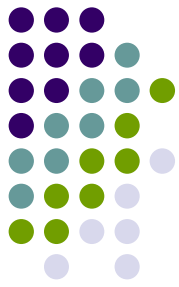
API & Callback functions



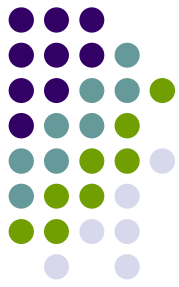
```
interface IServiceAPI {
    void publish(in String feedID, in Entry entry);
    void subscribe(in String feedID);
    void unsubscribe(in String feedID);
    void discover(in String selector);
    void undiscover();
    void registerCallback(IClientCallback cb);
    void unregisterCallback(IClientCallback cb);
}

oneway interface IClientCallback {
    void notify(in String feedID, in Entry entry);
    void discoveryNotify(in String availableContents);
}
```

Synchronization Manager

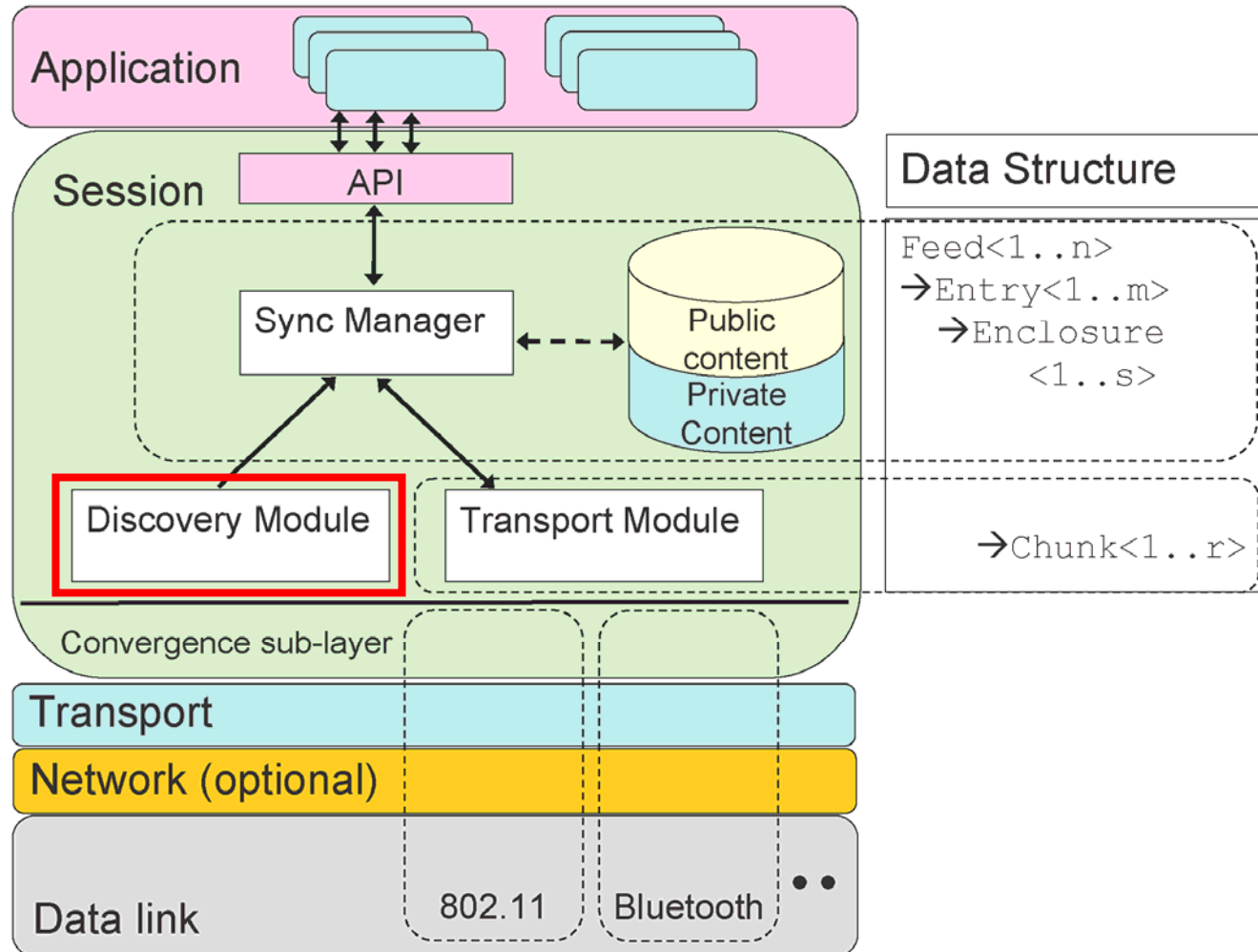
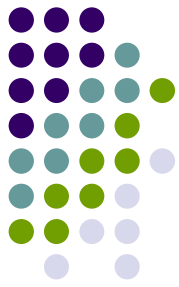


Synchronization Manager

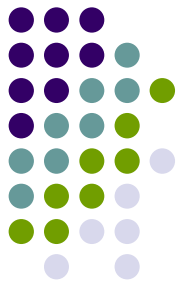


- Register and manage subscriptions
 - Process the API calls from applications
 - Initiates downloads on behalf of applications
- Provide a front-end to content database
 - SQLite Database with available feeds/entries
 - Synchronize access
 - We support multiple simultaneous downloads & uploads
- Content database implemented as an *Android Content Provider*
 - Android way of sharing data across applications
 - Audio, Video, Images, Contacts, phone calls, ...
 - No common storage area for all apps
 - Downloaded content available to all applications

Discovery Module

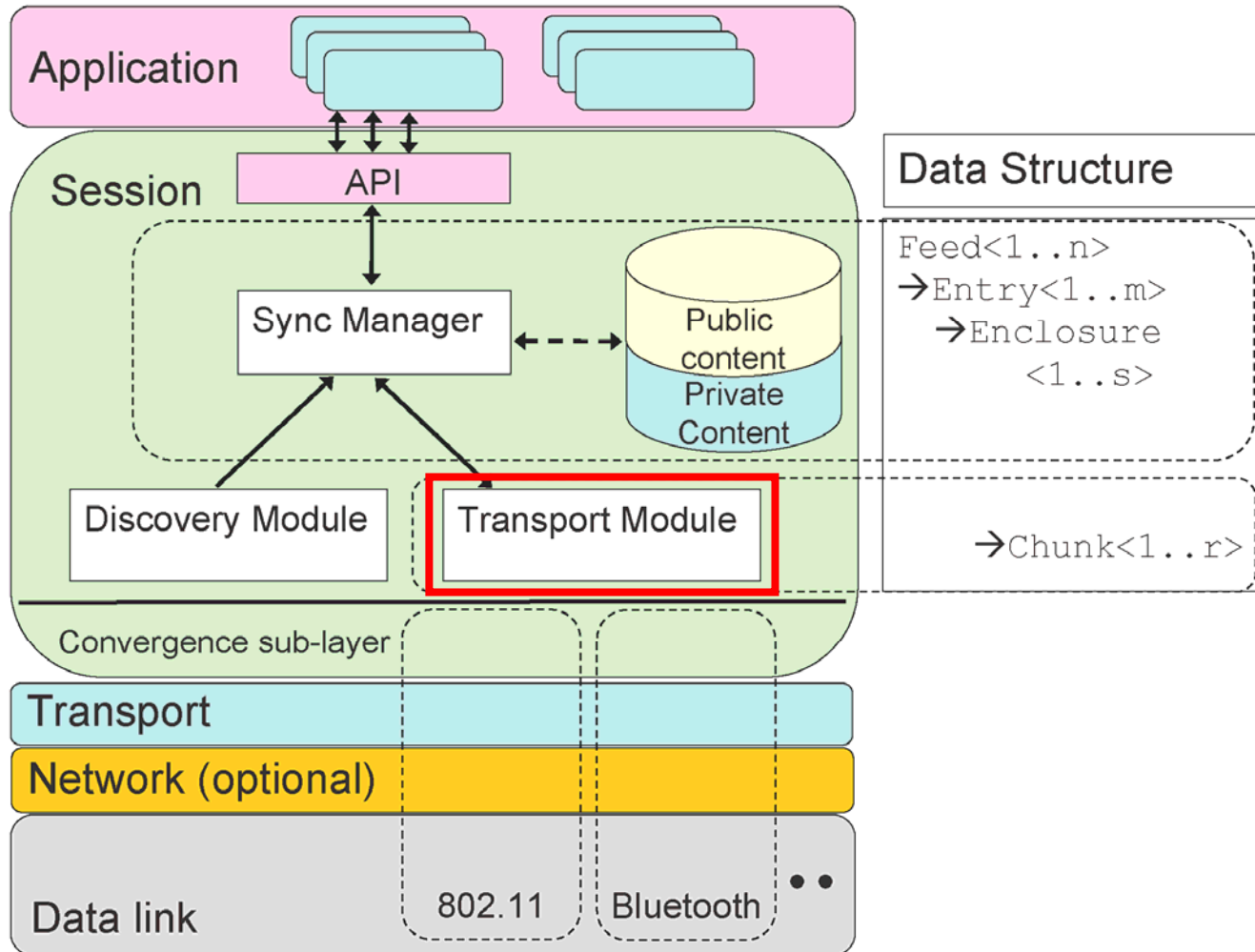
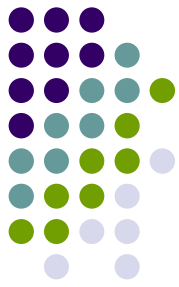


Discovery Module

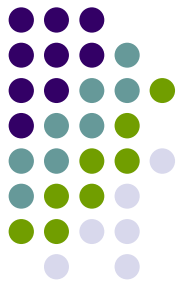


- Implementation based on 802.11 ad-hoc mode
 - Not yet supported in Android (not even 2.2!!)
 - Supported by Linux and device driver for wireless interface
 - Native code resets wireless interface
- Node/service discovery
 - Periodically send `hello` beacons
 - UDP broadcast datagram
 - Contains node ID and *content revision number*
 - Simple revision number for content database
 - Incremented when new content published or downloaded
- Associate with neighbor if
 - Never seen before
 - It has new content since last encounter
 - revision number higher than in contact history cache
 - There are new subscriptions locally

Transport Module



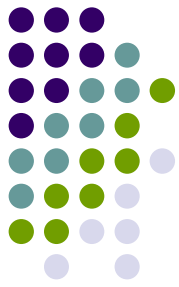
Transport Module



- Implements content solicitation protocol
 - Request/reply protocol
- Typical session alternates between *discovery* and *download* state
 - Discovery: Solicit meta-information
 - Download: Directly solicit entries
- Example:
 1. Ask for a list of available feed ID's at peer

```
<message version="0.1">
  <device-id>01:23:45:67:89:ab</device-id>
  <request>
    <feedlist></feedlist>
  </request>
</message>
```

Content Solicitation



- Example (continued):
 2. Receives feedlist
Compares with local subscriptions.
Sends request to discover new entries on a feed

```
<message version="0.1">  
  <device-id>01:23:45:67:89:ab</device-id>  
  <request>  
    <feed>  
      <id>tag:psvensson@gmail.com:android_only_blog</id>  
      <entrylist filter="author IN ('Peter', 'Olafur')"></entrylist>  
    </feed>  
  </request>  
</message>
```

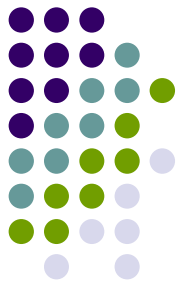
Content Solicitation



- Example (continued):
 3. Receives xml entry elements that match filter criteria
Send direct request for enclosure

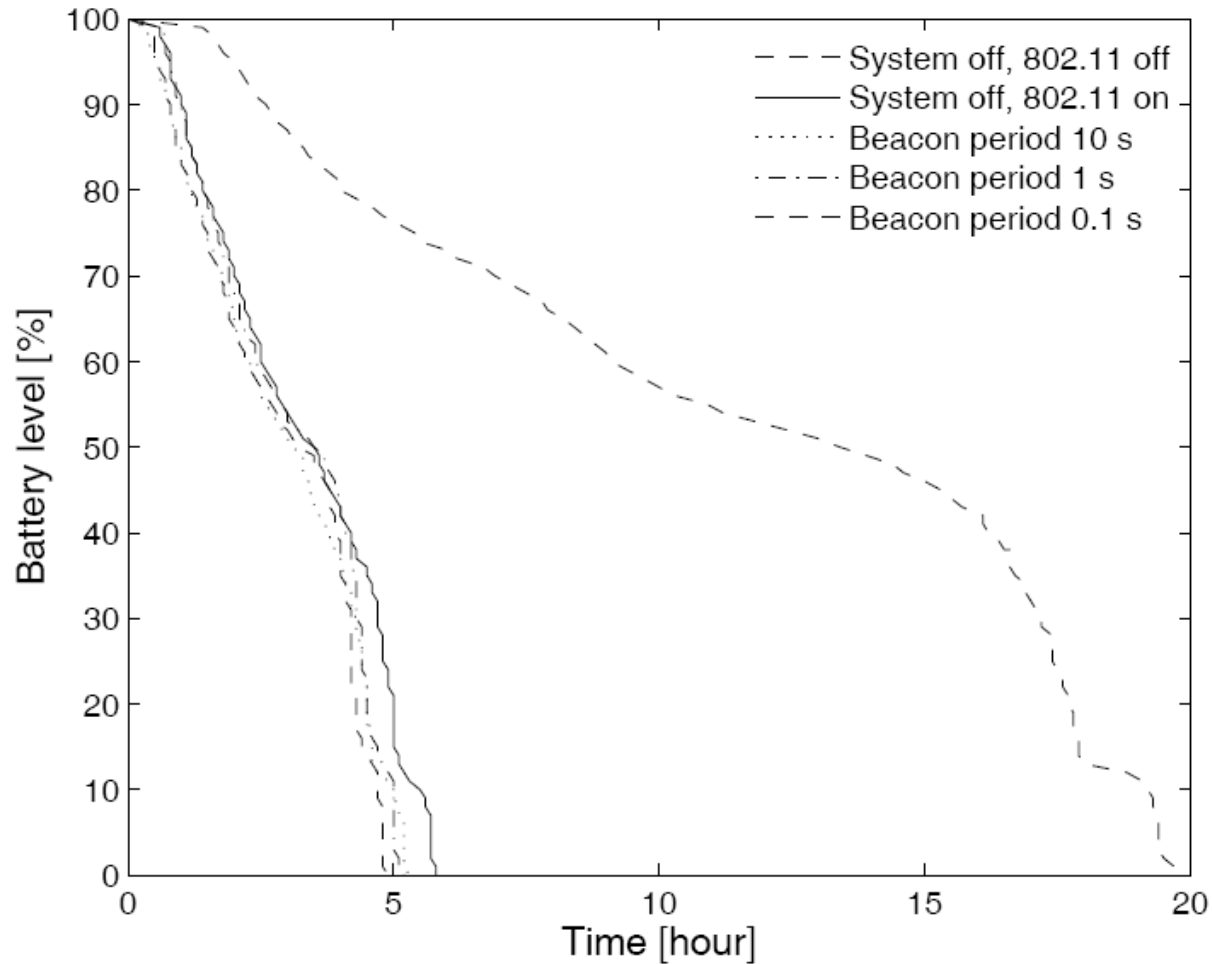
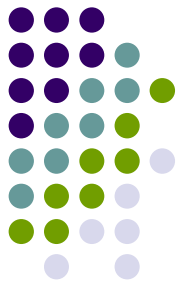
```
<message version="0.1">
  <device-id>01:23:45:67:89:ab</device-id>
  <request>
    <entry>
      <id>tag:psvensson@gmail.com:android_only_blog:Dancing_at_the_party</id>
      <link rel="enclosure"
        type="audio/mpeg"
        podnet:chunks="23-50"/>
    </entry>
  </request>
</message>
```

Transport Module

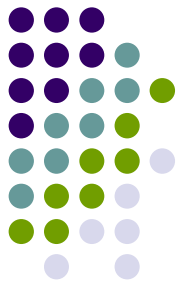


- Implementation uses TCP
- Server side and client side implemented separately
- Stateless server
 - Standard accept-“fork” pattern
 - “fork” a new thread to handle client
 - Max num concurrent clients configurable
- Client side also implemented in a Thread object
 - Discovery module starts a new Client when peer is discovered
 - Each node can have many active downloads simultaneously
- MAC-layer shares radio-link between sessions

Evaluation: Energy Consumption

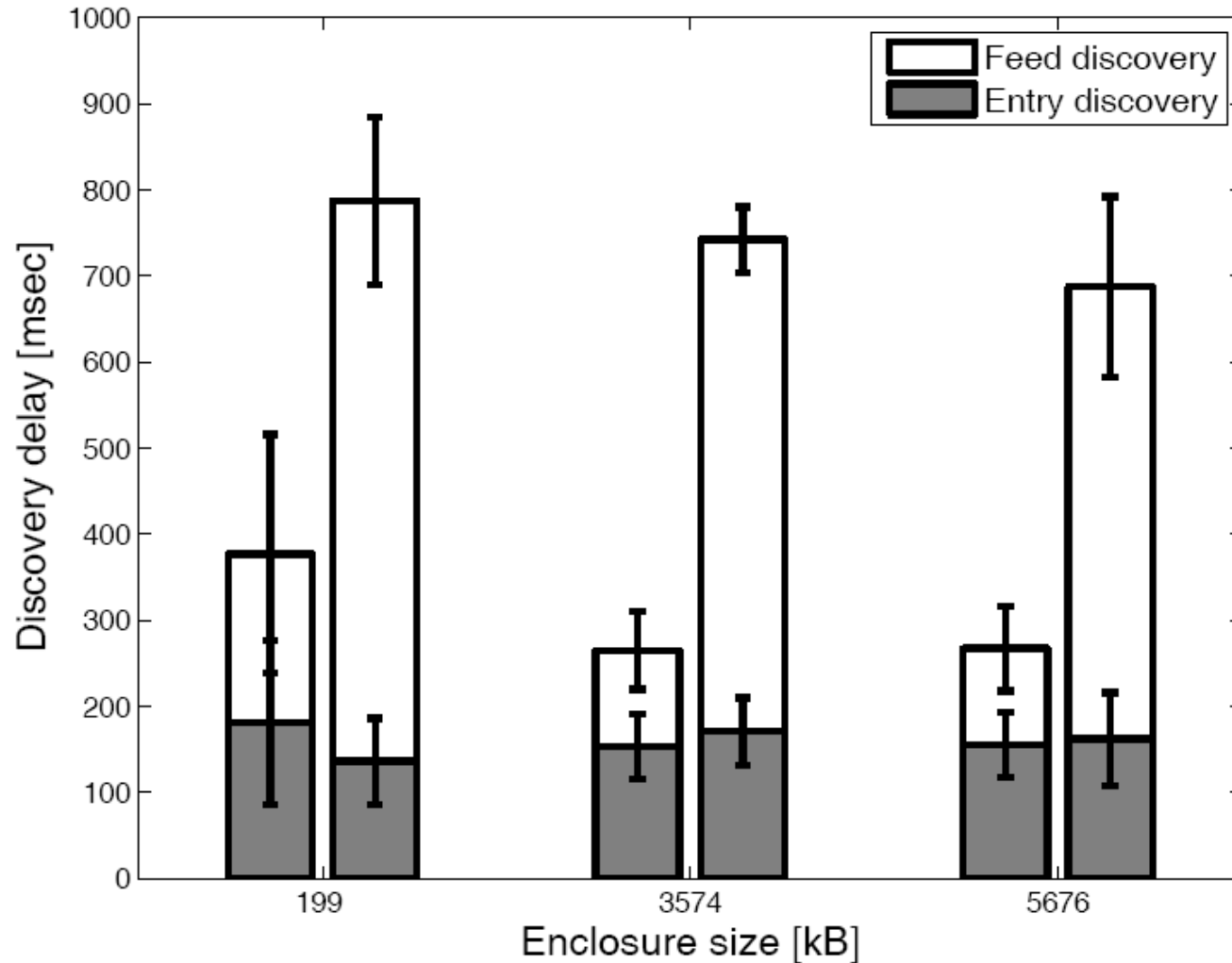
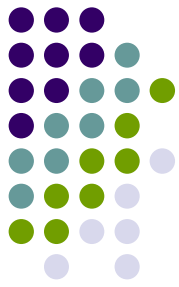


Profiling

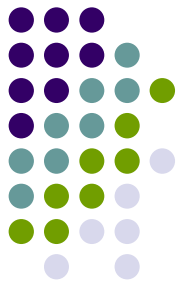


- Profile to verify correctness and assess performance
- Code instrumented with timestamps
 - Logging removed to minimize I/O
- Measure discovery phase for different enclosure sizes
 - Metric: Total discovery time
 - Feed discovery + Entry discovery

Profiling discovery time



Conclusion



- Implementation of middleware for opportunistic content distribution
- A best-effort opportunistic pub/sub system
 - General architecture that facilitates content-centric applications
- Based on Google Android platform
 - Powerful and quite mature platform
 - Some issues with 802.11
 - Ad-hoc mode not supported by framework
 - Power-hungry
- Future work
 - Content caching/forwarding
 - Power management
 - Security/privacy
- Demo app: Video/photo blogging
- More info: www.ee.kth.se/~olafurr