Network Layer:
Location/Service Management

Y. Richard Yang
11/27/2012

Admin.

- Project meeting slots posted on the Sign Up page of classesv2
  - If none of the time works, please send me email to schedule appointments
  - Each team needs to schedule two meetings to discuss your project by the end of semester
- Sharing of devices (status post to a google doc?)
  - 4 x 10" Android (2 Xoom, 1 Nexus 10, 1 Galaxy Tab 2 10.1)
  - 3 x 7" Android (2 Nexus 7/Wifi, 2 Nexus 7/Wifi/Cellular)
  - 1 x Android phone
  - 1 x iPad 2
  - 1 x iPad w/ Retina display
  - 1 x Nokia Lumia 920
  - 10 x GNURadio sets
  - 1 x Altera FPGA set

Recap: Improving Wireless Capacity

- A single half-duplex transceiver at each node
- Interference constraint
- Radio interface constraint

Recap: RRC State and App

- Given the large overhead for radio resource control (RRC), wireless networks implement RRC state machine on mobile devices for data connection

Case Study: Pandora Streaming

Problem: High resource overhead of periodic audience measurements (every 1 min)
Recommendation: Delay transfers and batch them with delay-sensitive transfers
Recap: Wireless Link Layer

- The basic services of the link layer
  - framing, link reliability, etc
  - link access: interference, quality of service (and fairness) control, link state management
- Guided by network layer
  - transmit to which neighbor at what quality

Guided by network layer:

- transmit to which neighbor at what quality

Outline

- Admin. and recap
- Network layer
  - Intro to networks

Network Layer Services

- Transport packets from source to dest
- Network layer protocol in host and router

Basic functions:

- Control plane
  - compute routing from sources to destinations
- Data plane: forwarding
  - move packets from input interface to appropriate output interface(s)

Network Layer: API

- API (provided to upper layer)
  - transmit(info, src, dest, ...)
- A key decision in network layer design is how to represent destinations?
  - we refer to how a client specifies destination as the addressing scheme
  - the supported addressing scheme(s) can have profound impacts on usability, flexibility, scalability

Discussion: How to Specify a Destination?

Two Basic Approaches for Identifying Destinations

- Locators
  - Encode locations on network topology
- Identifiers (ID)
  - Independent of network topology
Addressing Scheme: Telephone

- Very first scheme: connection by operators to business
  - Identifier or locator?

The telephone numbering scheme:

- invented in 1888 by Almon Strowger, an undertaker
  - “No longer will my competitor steal all my business just because his wife is a BELL operator.”

Addressing Scheme: Telephone

- E.164: Maximum 15 digits
- Hierarchical addressing scheme: country code + national destination (area) code (optional) + subscriber number
  - e.g., +1-203-432-6400

Why hierarchical addressing scheme?

Evolution of Telephone Addressing Scheme

- Identifier (business, person)
- Locator (hierarchical phone #)
- Identifier (person again w/ mobile phones, 800)

IP Addressing

- Also a hierarchical locator addressing scheme
  - network part (high order bits)
  - host part (low order bits)

What’s a network? (from IP address perspective)

Device interfaces with same network part of IP address

- link layer can reach each other

Addressing Scheme: Sensornet Example

- Destination: message to a sensor (e.g., who detected fire)
  - <ID = D> // id.
  - <Lat=37.3169; Long=-121.8740> // loc
  - <temperature = highest> // id
Addressing Scheme: Printer

- How may we specify the destination as the color printer on the 2nd floor of AKW
  - Internet domain name: lw2c.cs.yale.edu
  - Internet protocol (IP) address: 128.36.231.8
  - [building = AKW; floor=2; entity = printer; quality = color]

Routing in IP/Telephone Networks

- Represent a network as a graph
- Determine a path to each destination on the graph
- Key problems
  - Location management
    - Find attached point for id-based naming
  - Routing
    - Find path from src to dst attach point

The Service Discovery Problem

- How does a src connect to a dst in an ID-based addressing scheme?
  - ID-based phone#
  - IP address difficult to remember, uses domain name lw2.cs.yale.edu

Basic Network Layer Model

- Each node is a network attachment point (e.g., router, base station), to which hosts/user device attaches
- User device identified by addressing scheme
  - locator: identifies attachment point
  - identifier: independent of location

Outline

- Admin.
- Network layer
  - Intro to networks
  - Location management
    - service discovery

Today’s Internet: DNS

- Represent a network as a graph
- Determine a path to each destination on the graph
- Key problems
  - Location management
    - Find attached point for id-based naming
  - Routing
    - Find path from src to dst attach point
**DNS: Domain Name System**

- **Function**
  - Map domain name (e.g., cs.yale.edu) and service type (e.g., email) to IP address (e.g., 128.36.232.5).
  - Domain name: a hierarchical name space implemented by a distributed database to allow distributed, autonomous administration.

**DNS: Naming Scheme**

- Each DNS server stores resource records (RR).

  RR format: (name, value, type, ttl)

- **Type=A**
  - Name is hostname.
  - Value is IP address.

- **Type=NS**
  - Name is domain (e.g., yale.edu).
  - Value is the name of the authoritative name server for this domain.

- **Type=CNAME**
  - Name is an alias name for some "canonical" (the real) name.
  - Value is canonical name.

- **Type=MX**
  - Value is hostname of mail server associated with name.

**DNS Name Resolution**

- A host queries local DNS server, who may forward the query to the corresponding DNS server.

**Problems of Traditional DNS Service Discovery**

- Support fixed types of services, e.g., A, NS, CNAME, MX.
  - Static and fixed record structure makes it difficult to introduce new services or non-trivial service queries.

- Depends on infrastructure: a DNS server.

**General Naming Paradigm: Linda**

- "Distributed workspace" by David Gelernter in the 80's at Yale.

- Very influential in naming and resource discovery.

- Naming scheme:
  - Arbitrary tuples (heterogeneous-type vectors).

- Name resolution:
  - Nodes write into shared memory.
  - Node read matching tuples from shared memory.
  - Exact matching is required for extraction.
Linda: Core API

- out(): writes tuples to shared space
  - example: out(“abc”, 1.5, 12).
  - result: insert (“abc”, 1.5, 12) into space.
- read(): retrieves tuple copy matching arg list (blocking)
  - example: read(“abc”, A > B)
  - result: finds (“abc”, 1.5, 12) and sets local variables A = 1.5, B = 12. Tuple (“abc”, 1.5, 12) is still resident in space.
- in(): retrieves and deletes matching tuple from space (blocking)
  - example: same as above except (“abc”, 1.5, 12) is deleted.
- eval(expression): similar to out except that the tuple argument to eval is evaluated.
  - example: eval(“ab”, -6, abs(-6)) creates tuple ("ab", -6, 6).

Linda Extension: JavaSpaces

- Industry took Linda principles and made modifications
  - add transactions, leases, events
  - store Java objects instead of tuples
  - a very comprehensive service discovery system
- Definitive book, “JavaSpaces Principles, Patterns, and Practice”
  - 2 of 3 authors got Ph.D.’s from Yale.

JavaSpaces – Visual Overview

Progress

- Support fixed types of services, e.g., A, NS, CNAME, MX
- Linda name space
- Depends on infrastructure: a DNS server

DNS without Central DNS Server: mDNS

- Multicast in a small world
  - no central address server
  - each node is a responder
- link-local addressing
  - send to multicast address: 224.0.0.251
- Leverage DNS format—DNS-SD
  - each node picks own name:
    - <instance name>.
    - <app-proto>.
    - <service>.
    - <domain>
mDNS: Example
- Use the mDNS command on Mac as example
  - Advertise an LPR printer on port 515
    mDNS -R "My Test"_printer._tcp.515.pdl:application/postscript
  - Advertise a web page on local machine
    mDNS -R "My Test"_http._tcp.80.path=/path-to-page.html
- Browse web pages on local machines
  mDNS -B _http._tcp

Service Discovery in Android
- Based on mDNS/DNS-SD
- Foundation for peer-to-peer/Wi-Fi Direct in Android
- See http://developer.android.com/training/connect-devices-wirelessly/nsd.html

Service Registration in Android
```java
public void registerService(int port) {
    NsdServiceInfo serviceInfo = new NsdServiceInfo();
    serviceInfo.setServiceName("NsdChat");
    serviceInfo.setServiceType("_http._tcp.");
    serviceInfo.setPort(port);
    mNsdManager = Context.getSystemService(Context.NSD_SERVICE);
    mNsdManager.registerService(serviceInfo,
                                 NsdManager.PROTOCOL_DNS_SD,
                                 mRegistrationListener);
}
```

Service Discovery in Android
```java
mNsdManager.discoverServices(SERVICE_TYPE,
                             NsdManager.PROTOCOL_DNS_SD,
                             mDiscoveryListener);
```
```java
public void initializeDiscoveryListener() {
    // Instantiate a new DiscoveryListener
    mDiscoveryListener = new NsdManager.DiscoveryListener() {
        // Called as soon as service discovery begins.
        @Override
        public void onDiscoveryStarted(String regType) {
            Log.d(TAG, "Service discovery started");
        }

        @Override
        public void onServiceFound(NsdServiceInfo service) {
            // A service was found!  Do something with it.
            Log.d(TAG, "Service discovery success" + service);
            if (!service.getServiceType().equals(SERVICE_TYPE)) {
                // Service type is the string containing the protocol and
                // transport layer for this service.
                Log.d(TAG, "Unknown Service Type: "+ service.getServiceType());
            } else if (service.getServiceName().equals(mServiceName)) {
                // The name of the service tells the user what they'd be
                // connecting to. It could be "Bob's Chat App".
                Log.d(TAG, "Same machine: " + mServiceName);
            } else if (service.getServiceName().contains("NsdChat")) {
                mNsdManager.resolveService(service,
                                            mResolveListener);
            }
        }
    };
```

Challenge of using Name Resolution in Mobile Computing
- Name -> location binding can be dynamic as a mobile device changes attach point

Mobile Name-Loc Binding Design Issues
- Binding database update
- Hand off
  - dst may move during a session (phone call)
Outline

- Admin.
- Network layer
  - Intro to networks
  - Location/service management
    - service discovery
    - cellular network location management

Location Registry (LR) update in Cellular Networks

- Problem definition:
  - How to update the mapping between ph# and current attachment point (BTS) of the phone?

Two Primitives for Cellular Location Management

- Mobile station: reports to the network of the cell it is in
  - called update
  - uses the uplink channel

- Network: queries different cells to locate a mobile station
  - called paging
  - uses the downlink channel

Performance of the Two Primitives

- A city with 3M users
- During busy hour (11 am - noon)
- Update only
  - total # update messages: 25.84 millions
- Paging only
  - call arrival rate: 1433 calls/sec
  - total # paging transactions: 5.2 millions

Discussion

- A user receives one call for ~5 cells (25M vs 5M) visited, thus may not need to update after every switching of cell

- However, if no update at all, then paging cost can be high—may need to page the MS at every cell
  - Q: how do you page?
Location Management Through Location Areas (LA)

- Used in the current cellular networks
- A hybrid of paging and update
- Partitions the cells into location areas (LA)
- E.g., around 10 cells in diameter in current systems
- A global home location register (HLR) database for a carrier
- An MSC of each LA maintains a visitor location register (VLR) of the LA

LA Based Update/Paging

- Each cell (BTS) of an LA periodically announces its LA id
- If a MS moves to a new LA, it reports its location to visiting MSC
- MSC/VLR notifies HLR that it currently has MS
- When locating a MS, the network pages the cells in an LA

GSM

GSM Location Update: Example

- MS (mobile station)
- BSC (base station controller)
- BTS (base transceiver station)
- MSC (mobile switching center)
- GMSC (gateway MSC)
GSM Location Update: RR Connection Release

Remaining Issue: How to Decide the LAs: A Simple Model

- Assume the cells are given
- Cell \(1\) has on average \(N_1\) users in it during one unit time; each user receives \(\lambda\) calls per unit time
- There are \(N_{1j}\) users move from cell \(i\) to cell \(j\) in a unit of time

\[
\begin{align*}
\text{Cell 1} & \quad N_1 \\
\text{Cell 2} & \quad N_2 \\
\text{Cell 1} & \quad N_12 \\
\text{Cell 2} & \quad N_21
\end{align*}
\]

How to Decide the LAs: A Simple Scenario

- Separate LAs for cells 1 and 2
  - \#update: \(N_1 + N_2\)
  - \#paging: \(\lambda (N_1 + N_2)\)
- Merge cells 1 and 2 into a single LA
  - \#update: 0
  - \#paging: \(2\lambda (N_1 + N_2)\)

Cost Comparison

\[
C_{update}(N_{12} + N_{21}) = 2\lambda (N_1 + N_2)
\]

where \(C_{update}\) is relative cost of update to paging, assuming paging cost per cell is 1

- At the same mobility, if call arrival rate is high, more likely separate
- At the same call arrival rate, if higher mobility, more likely to merge

Extension: From GSM to GPRS to 3G UMTS

- Issue: it is anticipated that users will make more connections in data network
  - Same mobility but higher lambda \(\Rightarrow\) smaller location area
The LA/RA/UTRANA design considers:
- call pattern: when (how often) does a mobile station receive a call
- mobility model: how does a mobile station move

Issues of LA based approaches:
- Users roaming in LA borders may generate a lot of updates