# MobilityFirst: MSocket

By Alexander McRae (V00890516)

# What is MSocket?

- Userspace library
    - Java
    - Socket library
- Utilizes the MobilityFirst Internet Architecture
- Allows for peer mobility and multi homed peers
- Wraps TCP sockets

## Problem

- IP is being conflated with identity
- Modern workloads are requiring more mobility
- Current solutions to the problem either:
    - Handle only a subset of the situations
    - Require massive changes to current internet architectures

# What are the mobility situations?

- Connection time
  - What happens when a peer moves while being connected to?
- Individual
  - When a single peer moves network locations
  - Usually an IP address changes
- Simultaneous
  - When both peers move network locations at the same time

# What characteristics should a solution have?

- Handle all mobility situations
- Compatible with current internet technologies
  - NAT, DNS, HTTP, IP
- Scale well
  - Be able to handle “tomorrows” usage of the internet
- Maintain the end-to-end principle
- Provide a general solution rather than a specific one
  - Application specific solutions already exist and work well in constrained environments

# Previous Work

# An end-to-end approach to host mobility (2000)

- Uses DNS to update the IP address of a domain name
- Includes a token in an optional TCP header
- Token identifies the connection

Strengths:

- Compatible with existing technologies
- Handles mobility situations

Weaknesses

- Every mobile peer must have a domain name
- Updates take a while

Alex C. Snoeren and Hari Balakrishnan. 2000. An end-to-end approach to host mobility. In <i>Proceedings of the 6th annual international conference on Mobile computing and networking</i> (<i>MobiCom '00</i>). Association for Computing Machinery, New York, NY, USA, 155–166. DOI:https://doi.org/10.1145/345910.345938

# Mobile IP

- Effectively creates a proxy back to the original address after the peer changes IP addresses
- The other peer is unaware of the migration

Strengths

- Compatible with existing tech
- Handles all forms of peer mobility

Weaknesses

- Requires large changes to internet architecture
- Breaks end to end principle

https://www.wikiwand.com/en/Mobile_IP

# QUIC

- Each logical connection is given a connection ID
- Whenever a peer migrates it will probe a path to the other peer
- Onces a peer migrates it sends data and waits for a authentication challenge
- On success the connection is migrated

Strengths

- Defined security
- Handles most mobility situations present
- Compatible with existing tech

Weaknesses

- Simultaneous mobility not handled

https://www.ietf.org/archive/id/draft-ietf-quic-transport-34.html Sections 5 and 9

# MobilityFirst Overview

# History

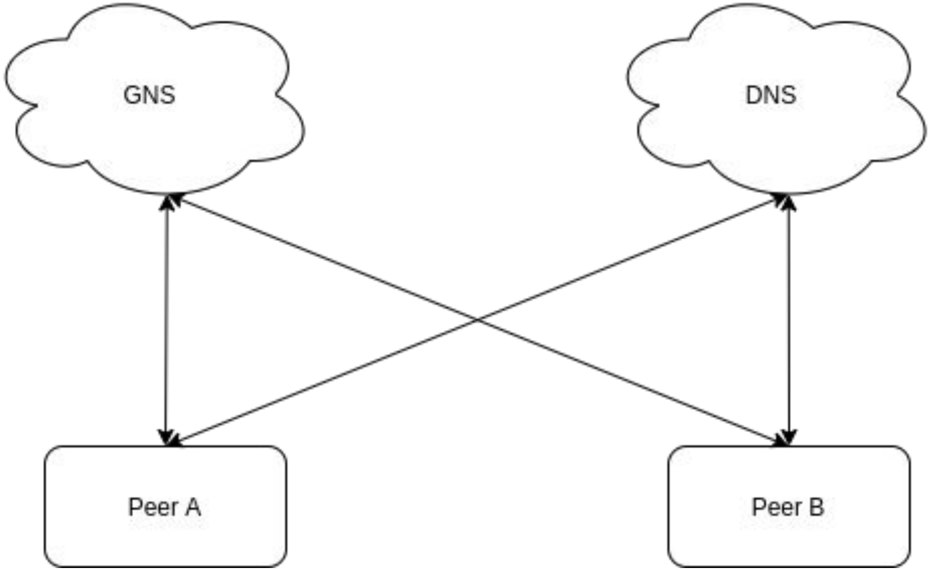
- Started in 2010 as part of the National Science Foundation's Future Internet Architecture program
- The idea was given a clean slate how would we change the internet
- Prototype as a realization of the stack
- Alongside it were
  - Named Data Networking
  - NEBULA
  - eXpressive
  - ChoiceNet

# Components of MobilityFirst (Relevant to MSocket)

- Separation of naming and addressing
  - GUID
- Global fast dynamic (updatable) name resolution service
  - GNS (Auspice)
- Self Certifying public private key authentication

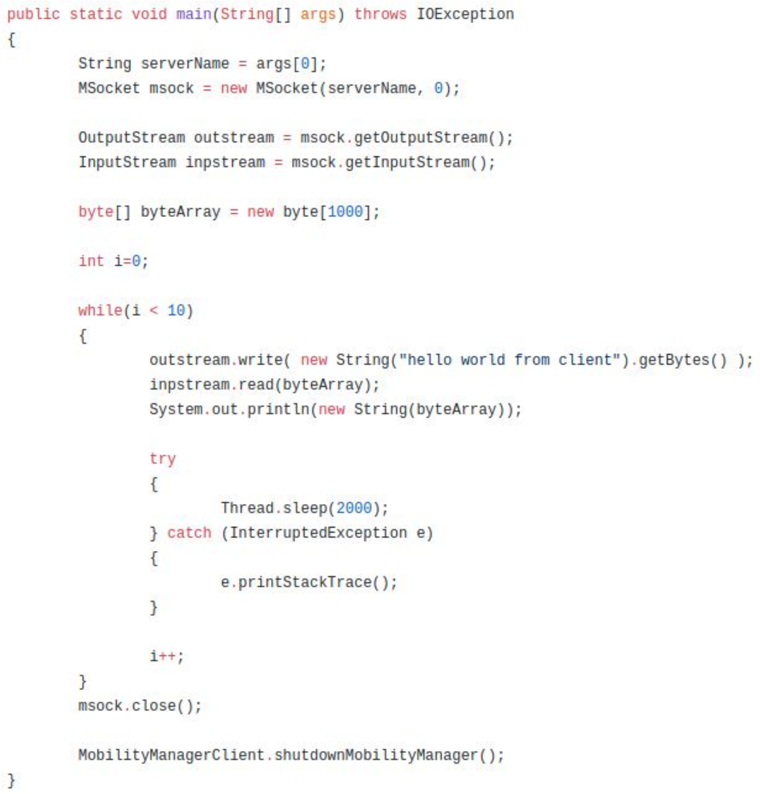
# Components Visualized

We can think of GNS as DNS but maps GUIDs to IP addresses rather than domain names to IP



# Interface

- Provides socket API similar to the BSD api
- Changes the interface slightly to be more JVM friendly



```
public static void main(String[] args) throws IOException
{
        String serverName = args[0];
        MSocket msock = new MSocket(serverName, 0);

        OutputStream outstream = msock.getOutputStream();
        InputStream inpstream = msock.getInputStream();

        byte[] byteArray = new byte[1000];

        int i=0;

        while(i < 10)
        {
                outstream.write( new String("hello world from client").getBytes() );
                inpstream.read(byteArray);
                System.out.println(new String(byteArray));

                try
                {
                        Thread.sleep(2000);
                } catch (InterruptedException e)
                {
                        e.printStackTrace();
                }

                i++;
        }
        msock.close();

        MobilityManagerClient.shutdownMobilityManager();
}
```

# How it works

- High level connection
  - identified by (Client ID, Server ID)
- Under MSocket is 1 or more TCP sockets
  - Identified by (Client IP, Client Port, Server IP, Server Port)
  - “Flow Paths”
- New packet format
  - Sits on top of TCP packet
- High level input and output buffers
- Connection Control Socket
  - Handle migration

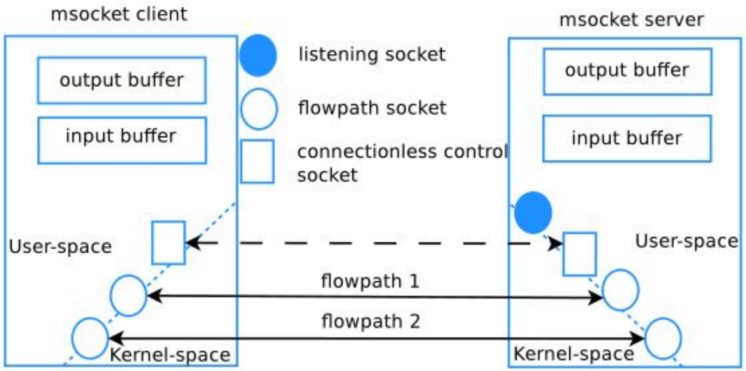


Figure 2: Overview of msocket components.

# MSocket Packet

- TCP Independent sequence number
- Cumulative ACK

- Needed because we are merging multiple TCP streams into a single output buffer

# Connection establishment

- The client will connect to the server using a TCP socket and establish
  - ConnID
  - PathID
  - Connectionless Control Socket Address (CCSA)
- If a peer moves during connection establishment, the GNS is contacted for the updated IP address

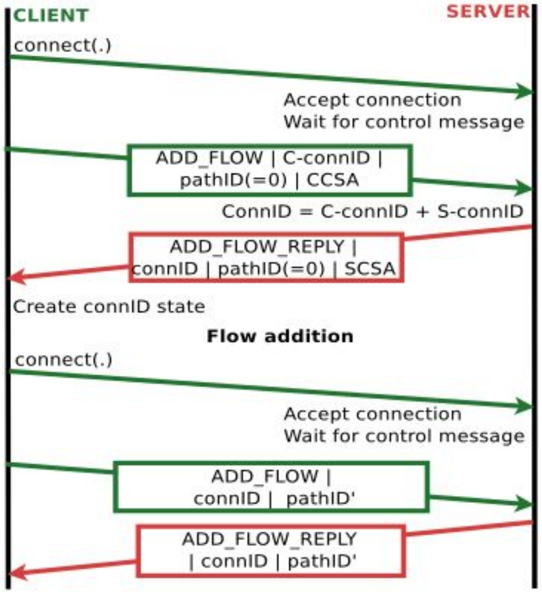


Figure 3: Connection establishment

# Data Transfer

- The data packets can travel over any flowpath
- Each packet contains at least
  - Sequence Number
  - Cumulative Acknowledgement
  - Length of payload
- The peer orders these correctly in the output buffer which the user can then see
- Normal packet loss in handled by the underlying TCP connections
- Since a single packet can travel along multiple flowpaths as long as there is a single connection TCP will deliver the packet

# Flow Migration

1. Close the underlying flowpath (TCP conn)
2. Opens a new flowpath
3. Server accepts but doesn’t know if this is a new connection, waits for control message
4. Sends the control message with ConnID and PathID to the CCSA
5. Server sends another control message to resend missing data

- Each flowpath can be migrated independently from each other
- A connection is migrated when at least 1 flowpath is re-established
- If the server wants to change locations it sends a reconnect control message telling the client where to reconnect
- If both move at the same time GNS is queried and then the client reconnects

## Strengths

- Only peers have to change internet socket library
- Handles all forms of mobility
- Compatible with NAT (not covered)

## Weaknesses

- Fails to stay compatible with normal TCP
- Not suitable for non-tcp like connections
- More of an solution for specific applications than general usage

# Conclusion

- The architecture and components set out by the MobilityFirst group are a fantastic start to making peer mobility on the internet happen
- The realization of the stack layed out does not meet the goals that I have for peer mobility

MSocket: https://web.cs.umass.edu/publication/details.php?id=2326

https://github.com/MobilityFirst/msocket