Investigating NDN Mobility Support

Tai-Lin Chu

Feb 14, 2015

1 Introduction

NDN mobility support is one of the sub-problem to make NDN REST possible. This is because a client or a server, which can be a data producer, can be mobile node.

2 Why is mobility hard in NDN?

- Fact 1: NDN node sends name availability at certain location (name, location), and routing protocol propagates this information throughout the network. As a result, if a node moves, this information is no longer useful. On the other hand, IP address itself represents location, and on top of IP, DNS resolves a particular name to a location. As a result, without frequent DNS update, a name's location needs to be static.
- Fact 2: NDN communication is asymmetric; that is, ideally, data should be put in data packet. On the other hand, all IP packets can carry data. (If NDN communication is symmetric, mobile producer won't be a problem if consumer is static.) The asymmetry communication gives mobility of consumer for free, but creates problems for mobility of producer (a producer needs to wait for routing update).

Note: *In IP, only static nodes use names, while mobile nodes use IP addresses (location)* for various reasons. And yet, we want all NDN nodes to use names.

3 Case summary

- Static producer/Static consumer: no issue
- Static producer/Mobile consumer: consumer re-sends interest

- Mobile producer/Static consumer: hard (can be solved by NDNS, KITE, pendingData)
- Mobile producer/Mobile consumer: very hard (can be solved by NDNS, KITE)

4 Current solution to mobility

- NDNS: create alias for a name in different subnets.
- KITE: use traced/tracing interest and static rendezvous anchor
- pendingData: leverage static consumer to fetch data from producer

5 Extending pendingData to Mobile producer/Mobile consumer

See my last memo for pendingData.

What is interesting about pendingData is that the producer does not need a name to publish data.

Here is the new loosen requirement to be a data producer under pendingData:

1. a prefix is assigned to you
2. or another node that has a prefix already needs your data (new)

This solution assumes there is a static node that serves as a trusted middle man, and pendingData is used as a sub-routine. Ideally the middle man should be a personal repo.

Producer – Middle Man – Consumer

- From Producer to Middle man: pendingData
- From Middle man to Consumer: consumer sends interest normally