Multi-Path Protocol for Big Data Transfer

Lotfi Benmohamed
NIST

Team: A. Battou, H. Bilil, O. El-Mimouni, K. Halba, C. Mahmoudi
Outline

• How can we move data faster by providing multi-path transfer capability through the network

• What can an ICN architecture like NDN offer (compared to IP)

• Show that it can be done easier and more efficiently with NDN than with IP

• Describe our implementation and its performance
Multi-path transfer

• Large data volume can come from either
  – small number of large data files (such as from scientific big data from previous talks), or
  – large number of small data objects (such as from video content distribution),
  – even though a large data volume in both cases

• To support big science data we’re interested in the problem of maximizing transfer rate (hence minimizing transfer time) when moving large data files

• Multi-path transfer uses multiple paths between two endpoints
  – network resources on all paths appear to the endpoints as a single pooled resource
  – dynamic scheduling (coordinated congestion control) used to split data traffic across the available paths

• Benefits to user
  – Higher throughput: due to pooled resources
  – Improved resilience to link or node failure (if we lose resources on one path we still have the resources of other paths)
Multi-path throughput

- With fully disjoint paths,
  - we can achieve $C = \sum_{i=1}^{n} C_i$

- With partially disjoint paths
  - If $C_0 \geq \sum_{i=1}^{n} C_i$, then as if fully disjoint (from throughput point of view)
  - Still useful when $\max(C_i) < C_0 < \sum_{i=1}^{n} C_i$, can still get some benefit from multi-path
How can we get multi-path capability

- [1] Network-controlled multi-path
  - No end-point involvement (end-user with single physical network connection)
  - Network splits end-user traffic into multiple paths

- [2] User-controlled multi-path through interface selection
  - End-point is multi-homed (with multiple physical access links)
  - Number of paths up to number of physical access links
  - No network involvement (user splits traffic)

  - User selects transit points (topologically diverse) for multi-path

Combinations are also possible
- [1]+[2]
- [1]+[3]
- [2]+[3]
- [1]+[2]+[3]
Network-controlled multi-path

• In IP networks, forwarding is always along the shortest path
  – If multiple such paths exist, equal-cost multi-path (ECMP) routing can be used for load balancing over them
  – To avoid IP routing loops only paths with the minimum routing cost can be considered
  – Limited number except in some regular & dense topologies (data centers)

• ECMP works at the flow level (5-tuple flow granularity)
  – but we need to split one big flow into sub-flows
  – multiple IP addresses per interface needed for multi-path tcp to work in this case (up to # ECMP paths)

• In NDN networks, multipath is easy and readily available
  – Through Interest-forwarding strategy (can split traffic arbitrarily)
  – Built-in loop detection mechanism allows exploring any path (not just shortest paths)
Our approach

- Our approach is based on transit node selection and we made an implementation based on this approach.
- Partly due to the fact that we started with the following problem:
  - Given large data object stored at customer location P, transfer it as fast as possible to cloud storage.
- Possible use case:
  - Customer logs in to the portal of its cloud service provider (CSP) to initiate transfer of a data object.
  - Provides the name of the data object /NIST/MML/archive1.
  - CSP selects a set of diverse locations and triggers each storage location to start downloading objects with the given name.
  - Wildcard character (/NIST/MML/archive1/*) meaning that it is ready to accept the next data packet without specifying any particular one [“exclude” option can be used instead].
    - P will reply with the next non-transmitted data packet.
  - For reliable delivery, the index of last received packet is appended to names in Interests.
  - Producer keeps track of lost Data packets for retransmission.
• Distributed Transfer Protocol (DTP)
  – each location will have a subset of the original data object
    (not necessarily a contiguous in-order block)
  – if needed, can be followed by another protocol to consolidate at location C
  – Can be adapted for Hadoop map/reduce application

• Next step: Multi-Path Transfer Protocol (MPTP)
  – performs end-to-end simultaneous transfers through a selected set of locations
  – simpler than DTP as only C needs to keep track of received/lost Data packets (P needs to do this in DTP)
NDN Multi-Path Transfer Protocol
ndnMPTP

- CSP instructs C to initiate simultaneous (parallel) transfers of the data file /NIST/MML/archive1 from producer P through a selected set of transit nodes T1, T2, …, Tn (with names denoted /T1, /T2, …, /Tn)
  - ndnMPTP running at each of the nodes
  - All /Ti names are reachable (advertised in routing)
ndnMPTP – step0

- First Interest generated at C with the name
  - /NIST/MML/ndnMPTP/archive1
- ndnMPTP running at P will return the first data block in a Data packet with a specification of FinalBlockID as part of the MetaInfo

Data ::= DATA-TLV TLV-LENGTH
  Name
  MetaInfo
  Content
  Signature

MetaInfo ::= META-INFO-TYPE TLV-LENGTH
  ContentType?
  FreshnessPeriod?
  FinalBlockId?
ndnMPTP – Interest

- Interests are generated at C for each Ti with the name
  - /Ti/ndnMPTP/NIST/MML/ndnMPTP/archive1/BlockID
- Consecutive BlockIDs are requested up to FinalBlockID
  - Subject to retransmission of lost blocks and flow control at C

- Interest processing at Ti
  - Send new Interest with name /NIST/MML/ndnMPTP/archive1/BlockID
- Interest processing at intermediate node
  - Standard NDN processing
ndnMPTP – Data

- For each Interest, deliver corresponding data block
- Name in Data packet is
  - /NIST/MML/ndnMPTP/archive1/BlockID

Data processing at intermediate nodes
- At Ti append /Ti/ndnMPTP to name:
  - /Ti/ndnMPTP/NIST/MML/ndnMPTP/archive1/BlockID
- Other nodes: normal NDN processing
### At Consumer:

<table>
<thead>
<tr>
<th>Event</th>
<th>Outstanding/inflight (per Ti)</th>
<th>Window (per Ti)</th>
<th>Round Trip Time (per Ti)</th>
</tr>
</thead>
<tbody>
<tr>
<td>On_Data (Rcv)</td>
<td>Oi=Oi-1</td>
<td>Wi=Wi+AIF/Wi (+1 per RTT)</td>
<td>RTT_i=T_sent − T_current</td>
</tr>
<tr>
<td>On_Interest (Send)</td>
<td>Oi=Oi+1</td>
<td></td>
<td>Set Timeout for Interest</td>
</tr>
<tr>
<td>On_Timeout (no Data within TOi)</td>
<td>Oi=Oi-1</td>
<td>Wi=Wi*MDF (1/2)</td>
<td>Retransmit Interest</td>
</tr>
</tbody>
</table>

#### Per Ti:
- **Initialize TO**
  - subsec. up to 1 sec
- **When first RTT is measured set**:
  - RTTave=RTT
  - RTTdev=RTT/2
- **Whenever a subsequent RTT is collected set**:
  - RTTdev=b*|RTTave-RTT| + (1-b)*RTTdev [1/8]
  - RTTave= a*RTT + (1-a)*RTTave [1/4]
- **When sending Interest after first RTT measurement**
  - TO= RTTave + K*RTTdev [4]
Evaluation

- Preliminary simulations using ndnSIM
- Implementation using NDN-Cxx libraries
- Still ongoing
Next steps

• Did not yet activate multi-path feature in the NDN strategy
  – To find out how it will change the dynamics of TC
  – Expected to provide more benefit through additional load balancing

• NDN multipath congestion control
  – Better understanding (beyond single path)

• Testbed evaluation
  – Code on Github
  – NIST gateway node on NDN testbed
  – Plan to test/collect traces on testbed