Server-side performance evaluation of NDN

Xavier MARCHAL, Thibault CHOLEZ, Olivier FESTOR

LORIA, UMR 7503 (University of Lorraine, CNRS, INRIA)
Vandoeuvre-les-Nancy, F-54506, France

ACM ICN 2016, Kyoto, Japan
Outline

1. Introduction
2. Experimental environment
3. Server-side performance evaluation
4. Reduce the signature impact
5. Conclusion
<table>
<thead>
<tr>
<th></th>
<th>Outline</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Introduction</td>
</tr>
<tr>
<td>2</td>
<td>Experimental environment</td>
</tr>
<tr>
<td>3</td>
<td>Server-side performance evaluation</td>
</tr>
<tr>
<td>4</td>
<td>Reduce the signature impact</td>
</tr>
<tr>
<td>5</td>
<td>Conclusion</td>
</tr>
</tbody>
</table>
Context

Named-Data networking

Optimized for massive content diffusion:
- In-network cache and Interest concatenation

Rely on digital signature for content authentication:
- Each Data packet must be signed
- Compute SHA-256 then sign with a cryptographic algorithm

Problem statement

Actual cost when a content provider generates Data packets?
- Can be critical for real-time applications that must deal with unpredictable flows of data (ex: video live streaming, ...)

ACM ICN 2016
Server-side performance evaluation of NDN
Outline

1. Introduction

2. Experimental environment
   - Ndnperf
   - Testbed

3. Server-side performance evaluation

4. Reduce the signature impact

5. Conclusion
We developed ndnperf

A performance evaluation tool

Available in C++ and Java

Features:
- Report throughput, latency and packet processing time
- Multi-threaded application
- Generate Data packets from random data or real files
- Can generate asymmetric keys thanks to the NDN library

http://madynes.loria.fr/software/ndnperf_cpp.zip
Testbed

Specifications

2 servers:

- CPU: 2x8 cores E5-2630v3
- RAM: 64GB DDR4
- NIC: Intel X540
- OS: Ubuntu 15.04 (3.19)

Network

- Physical layer: 10GBASE-T (ad-hoc network)
- NDN network: 2 nodes, 1 NFD (0.4.0) per server
## Outline

1. Introduction

2. Experimental environment

3. Server-side performance evaluation
   - Single-Thread environment
   - Cache performance
   - Multi-Thread server

4. Reduce the signature impact

5. Conclusion
Single-Thread environment

Signing process

Like the available NDN tools

Interpretations

NFD limits SHA-256 throughput:

- Decent performance:
  - 500Mbps with 8K payload
- But no authentication

The server limits RSA throughput:

- 34Mbps with 8K payload
Cache performance

Pre-condition
Caches are preloaded with Data

Results
Cache throughput is much faster:
- Up to 1.8Gbps for local cache
- But 670Mbps for distant cache
When local node forwards packets:
- Additional PIT/FIB lookup
- Forwarding has a high cost

Throughput (packet/s) vs Payload (octet)
- Local cache
- Distant cache

%CPU vs Payload (octet)
- Client
- NFD client
- NFD server
Multi-Threaded producer

Observations

Adding threads increases throughput:
- x6.7 with 8 signing threads
- x11.6 with all logical cores

Can saturate NFD with 25 threads:
- 400Mbps with 8K payload

Heavy load for Data generation
⇒ Problem for real-time applications
⇒ Possible improvements?
Outline

1. Introduction
2. Experimental environment
3. Server-side performance evaluation
4. Reduce the signature impact
   - Improve signing function
   - Change default signing algorithm
   - Increase packet size
   - Improvements vs default
5. Conclusion
**Improve signing function**

**Statement**

For each Data packet:

- Find key pair
- Compute $\text{SignatureInfo}$
- 2 hashes computations
- Load public and private keys

⇒ Can be done once and for all

**Speed up**

13% for RSA or 20% for ECDSA
Available asymmetric algorithms

1. **RSA(1024, 2048)**
2. **ECDSA(256, 384)**

### Interpretations

- Lowering the key size can help tiny servers to serve more users.
- ECDSA-256 seems to be a better solution for servers since it has comparable security lvl (but is harder to verify for clients).
Increase packet size

**Hypothesis**

- **Current limit size:** 8800 Bytes
- **Fewer signatures per content:**
  - Reduce computation cost
  - Interesting for big contents

**Consequence**

- Speed up NDN throughput
  - 900Mbps with 32k payload
Great throughput increase per core:
- From 30 to 200 Mbps/core (x6.7)

Need less cores to saturate NFD:
- Only 5 cores against 14 for default RSA (9 for ECDSA-256)
Outline

1. Introduction
2. Experimental environment
3. Server-side performance evaluation
4. Reduce the signature impact
5. Conclusion
Conclusion

In these experiments we demonstrate that

- NFD cache performs good throughput
- Servers may have difficulties to serve users for real-time apps
  - No problem with SHA-256, but no authentication
  - Heavy load when using asymmetric algorithms (sum up below)

Improvements can be done by

- Changing algorithms
- Improving the signing function
- Increasing the packet size

<table>
<thead>
<tr>
<th>Source of NDN Data</th>
<th>Throughput</th>
<th>Nb of server cores</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client-side cache</td>
<td>1792 Mbps</td>
<td>0</td>
</tr>
<tr>
<td>Server-side cache</td>
<td>671 Mbps</td>
<td>0</td>
</tr>
<tr>
<td>Server with SHA256</td>
<td>487 Mbps</td>
<td>1</td>
</tr>
<tr>
<td>Server with RSA-1024</td>
<td>460 Mbps</td>
<td>6</td>
</tr>
<tr>
<td>Server with RSA-2048</td>
<td>394 Mbps</td>
<td>14</td>
</tr>
<tr>
<td>Server with ECDSA-256</td>
<td>439 Mbps</td>
<td>9</td>
</tr>
<tr>
<td>Server with ECDSA-384</td>
<td>397 Mbps</td>
<td>14</td>
</tr>
<tr>
<td>Server with improvements</td>
<td>922 Mbps</td>
<td>5</td>
</tr>
</tbody>
</table>