Experience and Evaluation of the Distributed Node Consensus Protocol
draft-jin-homenet-dncp-experience-00

Abstract

   The Distributed Node Consensus Protocol (DNCP) is a generic state
   synchronization protocol that offers data synchronization and dynamic
   network topology discovery within a network. This document reports
   experience with the DNCP, which includes its implementation status
   and performance evaluation in a simulated environment.

Status of this Memo

   This Internet-Draft is submitted in full conformance with the
   provisions of BCP 78 and BCP 79.

   Internet-Drafts are working documents of the Internet Engineering
   Task Force (IETF). Note that other groups may also distribute
   working documents as Internet-Drafts. The list of current Internet-
   Drafts is at http://datatracker.ietf.org/drafts/current/.

   Internet-Drafts are draft documents valid for a maximum of six months
   and may be updated, replaced, or obsoleted by other documents at any
   time. It is inappropriate to use Internet-Drafts as reference
   material or to cite them other than as "work in progress."

   This Internet-Draft will expire on January 7, 2016.

Table of Contents

1. Introduction

   The Distributed Node Consensus Protocol (DNCP)
   [I-D.ietf-homenet-dncp] is a protocol providing node data
   synchronization and dynamic network topology discovery within a
   network. At the time of writing this document, DNCP is in
   standardization process by the IETF Homenet working group.

   In DNCP, the information of a node and its view of the network is
   encoded in a set of TLV (Type-Length-Value) tuples. By publishing
   and exchanging the TLVs with its neighbors, each node in the network
   eventually receives the set of TLV published by every other node
   within the network (in which case, the network is converged). The
   Trickle algorithm [RFC6206] is used, instead of periodic signaling,
   to reduce the overhead of synchronization. DNCP also provides an
   option of "keep alive" message to detect when a neighbor is not
   reachable anymore.

   As a generic protocol, DNCP can not only be applied in home networks,
   but also in other networks that require node data synchronization
   (such as configuration profiles, network topology, services, etc.).
   Therefore, DNCP leaves some parameters to be specified by DNCP
   profiles, which are actual implementable instances of DNCP. Nodes
   implementing the same DNCP profile, and operating within the same
   DNCP network, are able share TLV tuples, discover the network
   topology, and auto-detect arrival and departure of other nodes.

   This document presents experience and performances evaluation using
   the reference DNCP implementation in a simulated environment using
   the Network Simulator 3 (NS3), a discrete event simulator widely used
   in network research and recognized by the scientific community.

   Note that for the purpose of this first study, DNCP was evaluated in
   various, quite unrealistic and probably extreme, network topologies,
   with the intent of finding the limits of the protocol.

2. Implementation Status

   Until July 2015, there are 2 publicly known implementations of DNCP,
   one of which was recently modified.

2.1. hnetd Implementation (libdncp)

   'hnetd' is an open source implementation of the Homenet network
   stack. As an implementation of the Home Network Control Protocol -
   HNCP [I-D.ietf-homenet-hncp], it includes the DNCP, the prefix
   assignment algorithm [I-D.ietf-homenet-prefix-assignment], as well as

other elements specific to Homenet. hnetd is available as a package
to be installed on OpenWrt, which is the platform it is most suited
for (hnetd also runs in a lesser extend on standard Linux). The code
is available on github [1] and is published under the GPLv2 license
(libdncp is available in the 'libdncp' git branch).

hnetd is based on a lightweight toolkit library containing useful
structures (e.g., lists, trees), functions (e.g., md5), as well as a
small event loop, that is widely used in OpenWrt. Thanks to the
quality of the code, libdncp, as separately buildable library
implementing DNCP functions, was easily extracted from hnetd for the
purpose of this work.

hnetd, DNCP included, consists of 15651 lines of code (18220 when
including test files). The binary weights 576KB when compiled for
debian X86_64 with no optimization and 727KB when compiled for
OpenWrt MIPS. libdncp2 roughly consists of 2300 lines of code (2590
when including security option). It weights 193KB when compiled with
no optimization for debian x86_64 and 192KB when compiled for OpenWrt
MIPS.

2.2. libdncp2 Implementation

As a result of different interests expressed about using DNCP outside
of Homenet (including this study), DNCP code within hnetd was partly
rewritten and reorganized in a more independent implementation of
DNCP, with less coupling with HNCP [I-D.ietf-homenet-hncp]. libdncp2
moves the DNCP profiles specificities from compilation-time to run-
time. It is published under the same license as hnetd and is now
part of the mainstream branch [1].

libdncp2 provides some improvements with respect to its predecessor
libdncp. For the purpose of this document, libdncp was used, but we
plan to use libdncp2 instead in the next iterations of this document.

2.3. shncpd Implementation

shncpd is an open source implementation of DNCP developed by Juliusz
Chroboczek. It uses HNCP [I-D.ietf-homenet-hncp] profile. The
source code is available on github [2]. At the time of publishing
this document, shncpd implements:

o The HNCP profile of DNCP.

o Parsing of a significant subset of HNCP.

o Address allocation (not prefix allocation).

3. Experimental Setup

   This section describes the environment, the parameters and topologies
   which were used for measuring DNCP performances.

3.1. Simulation Environment

   For the purpose of this work, the DNCP part (libdncp) of 'hnetd'
   introduced in Section 2.1 was modified in order to provide a
   statically linkable library containing DNCP implementation. To
   evaluate the performance of DNCP in large and complex networks, NS3
   is employed as the simulation platform.

   As shown in Figure 1, all the application-level actions (processing
   packets, publishing TLVs...) are performed inside the libdncp
   implementation. The packets are sent to or received from the lower
   layers in ns3 through the redefined socket API. UDP is used for
   layer 4 and IPv6 for layer 3. NS3 implements different types of
   links as layer 1 and layer 2. For these measures, the so called
   'CSMA' link type is used. The NS3 CSMA link is designed in the
   spirit of Ethernet but implements fail-proof carrier sense used for
   collision avoidance. For this first study, where measuring DNCP link
   usage was desired, link of virtually infinite throughput are used.

```
       +--------------------------------------+
       |     dncp implementation (Libdncp)    |
       +--------------------------------------+
       +--------------socket API--------------+
       |    +--------------------+            |
       |    |      L4 (UDP)      |            |
       |    +--------------------+            |
       |                                      |
       |    +--------------------+            |
       |    |      L3 (Ipv6)     |     NS3    |
       |    +--------------------+            |
       |                                      |
       |    +--------------------+            |
       |    |L1 and L2 (CsmaModel)|           |
       |    +--------------------+            |
       +--------------------------------------+
```
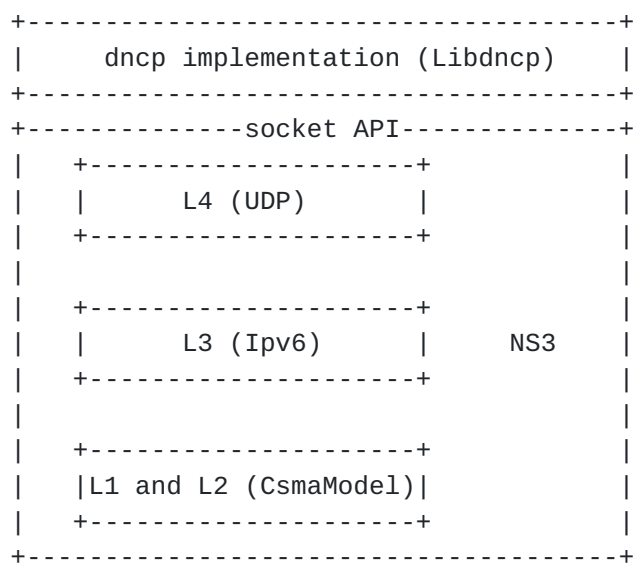
           Figure 1: libdncp over NS3 network layering model

   Listed below are several attributes of the CSMA model:

MTU: The link layer maximum transmission unit, set to 1500.

Encapsulation Mode: Type of link layer encapsulation. "Dix" mode
   was used, which models an Ethernet II layer.

TxQueue: Type of the transmit queue used by the device. NS3
   provides "Codel queue", "Drop tail" and "RED" (random early
   detection) queues. "Drop tail" queue was used with a size of 100
   packets.

Inter-frame Gap: The pause between two frames, set to 0.

Listed below are several attributes of the CSMA channel:

Data Rate: Physical layer bit rate, enforcing the time it takes for
   a frame to be transmitted. It is set to 1Gbps, which is
   significantly greater than what DNCP is expected to use.

Delay: Signal propagation time within the medium. It was set to 1
   micro second.

Assuming a constant frame size, the theoretical throughput of the
medium is given by the formula FrameSize/(FrameSize/DataRate +
Delay). For a frame size of 1500 bytes, the throughput is 923Mbps.
For a FrameSize of 100 bytes, the link throughput is 444Mbps.

Running DNCP in NS3 requires libdncp to be integrated and built with
NS3. DNCP runs on an event loop managed by libubox, which therefore
specifies how to set timers and listen to file descriptors.
Integration was quite straitforward. Event-loop and system calls
were identified and replaced with their NS3 equivalents. Besides, an
application in ns3 called "DncpApplication" is created, this
application can be installed on the node and can be started and
stopped at given time. Once the application is launched, dncp begins
to run on that node by calling functions in the libdncp static
library from inside the ns3 application. It is expected that
integration with libdncp2 will be even simpler, as this new library
put the DNCP profile definition at runtime instead of compilation
time.

Running experiments in simulated environments offers multiple
advantages such as the ability to run long-lived scenarios in short
period of time, simulate networks of hundread of nodes without
requiring lots of resources, or isolate tested components from
external interferences. On the other hand, NS3 executes program
steps virtually instantaneously, it is therefore hard to take into
account hardware speed when measuring time-related performances
metrics. For these first experiments, virtually perfect links with

no delay were used, and a processing delay of 0.5ms for each received
packet was introduced in order to simulate the packet processing
time.

3.2. Performance Metrics

The goal of this study is to measure the performances of DNCP in some
extreme scenarios, see whether, and how it converges. Therefore, the
following three performance metrics were observed:

Convergence time: The time it took for the network to converge.

Overall traffic sent: The amount of data that was sent on link
   before convergence.

Average traffic sent per node: The overall traffic sent divided by
   the number of nodes.

3.3. Chosen Toplogies

This section describes different topologies used in this study. The
main criteria for selecting a topology was that it should be:

o Easily described and generated as a function of the number of
   nodes (called N).

o Deterministic.

o Representing different situations testing different scalability
   properties.

The rest of this section describes the five different topologies:

o Single link

o String

o Full mesh

o Tree topology

o Double tree topology

3.3.1. Single Link Topology

The single link topology puts all the nodes on the same link. Each
node has a single DNCP End-Point with N-1 neighbors. Such topology
is well suited for evaluating DNCP scalability capabilities in terms

of the number of neighbors on a given link.

```
           n1        n2        n3        n4
           |         |         |         |
           +--------+--------+--------+
```
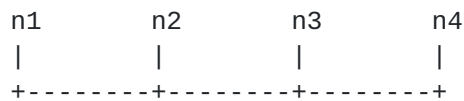
Figure 2: The single link topology for N = 4

3.3.2. String Topology

The string topology chains all nodes one after the other. Each node
has two DNCP End-Points with one neighbor on each (except for the two
extremities). Such topology is well suited for evaluating the
convergence time depending on the diameter of the network as well as
the scalability of DNCP in terms of the overall number of nodes.

```
           +-------+          +-------+
    |       |       |       |       |
    n1      n2      n3      n4      n5      n6
    |       |       |       |       |       |
    +-------+          +-------+          +-------+
```
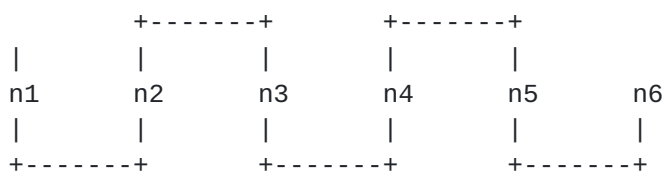
Figure 3: The string topology for N = 6

3.3.3. Full Mesh Topology

The mesh topology connects all nodes with distinct links. Each node
has N-1 DNCP End-Points with one neighbor on each. Such topology is
well suited for evaluating DNCP scalability capabilities in terms of
the amount of nodes and end-points.

```
           n1----n2--+
           | \    |    |
           |  \   |    |
           |   \  |    |
           |    \ |    |
           |     \|    |
           n3----n4   |
           |          |
           +---------+
```
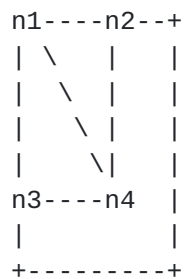
Figure 4: The full mesh topology for N = 4

3.3.4. Tree Topology

The tree topology forms a typical binary tree. Node 'i' is connected
with nodes '2*i' and '2*i + 1', unless those numbers are greater than
N. In such topology, all nodes except the root one have three DNCP
End-Points with one neighbor on each. This topology offers a more
realistic equilibrium between the diameter and the amount of nodes.

```
            n1
           /  \
         n2    n3
        / |    |
       /  |    |
      n4  n5  n6
```
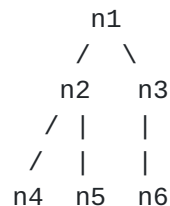
          Figure 5: The tree topology for N = 6

3.3.5. Double Tree Topology

The double tree topology is identical to the binary tree, but each
node is paired with a redundancy node. In such topology, all nodes
except the two root node have 6 DNCP End-Point with one neighbor on
each. This topology also offers a more realistic trade-off between
the network diameter and the number of nodes, but also adds
redundancy and loops.

For example, for N = 9:

o n1 has point-to-point links with n3, n4, n5 and n6.

o n2 has point-to-point links with n3, n4, n5 and n6.

o n3 has additional point-to-point links with n7, n8 and n9.

o n4 has additional point-to-point links with n7, n8 and n9.

4. DNCP Performance Evaluation

This section provides different performance metrics for different
network topologies of different sizes. Each value is the average
over 10 simulations. Unless stated otherwise, the 10 measures always
were pretty close (Small standard deviation).

Each simulation reflects the same scenario. All DNCP instances are
simultaneously initialized. The simulation is then run until the
network converges, and performance metrics introduced in Section 3.2

are computed.

Important: Note that, as DNCP uses Trickle, it is expected to be very
verbose in case of dramatic changes, but as a trade-off should
provide good convergence times and low verbosity in the absence of
changes. An instantaneous and complete bootstrap of the network,
which is the present scenario, is a particularly extreme state
change. This scenario was selected as a worst case to see whether
DNCP converges well, or at-all. On that side, DNCP behaved well, but
as expected, the overall amount of generated traffic sometimes
appeared to be significant.

4.1. Results for the Single Link Topology

Number of nodes	10 nodes	20 nodes	30 nodes	40 nodes
Convergence time	1.84s	3.09s	*4.43s	5.14s
----	----	----	----	----
Traffic sent	85.3KB	604.7KB	2.3MB	5.4MB
----	----	----	----	----
Traffic sent per node	8.5KB	30.2KB	79.6KB	140.7KB

                   Table 1: Single Link Topology (1)

Number of nodes	50 nodes	60 nodes	70 nodes	80 nodes
Convergence time	6.53s	*8.61s	11.57s	14.05s
----	----	----	----	----
Traffic sent	11.9MB	23.7MB	51.7MB	88.1MB
----	----	----	----	----
Traffic sent per node	245.KB	404.8KB	757.2KB	1.1MB

*: the average value was calculated over the results of 9 experiments
        because one of the value was significantly greater.

                   Table 2: Single Link Topology (2)

Note that two accidents were observed during the simulations. One
happened in one experiment among the 10 that we ran for the 30-nodes
network. The network first converged at 4.016s, which is very close
to the average convergence time, but at 25.949s this converging state
was broken and the network finally reconverged at 26.12s. Similarly,
for the 60-node network, it first converged at 7.081s then got
disturbed at 25.822s and converged again at 26.303.

   Although the convergence time seems to grow linearly with the number
   of nodes, the overall traffic sent before convergence increases
   dramatically. This is caused by the fact that not-only each node
   will synchronize with any other node on the link, but the data will
   grow as nodes discover their neighbors. The problem is explained in
   [I-D.ietf-homenet-dncp] Section 6.2 (Support For Dense Broadcast
   Links) where an optional solution is also proposed, but that option
   was not implemented in libdncp, which explains the bad
   performances.

4.2. Results for the String Topology

   +------------------------+----------+----------+----------+----------+
   | Number of nodes | 10 nodes | 20 nodes | 30 nodes | 40 nodes |
   +------------------------+----------+----------+----------+----------+
Convergence time	1.84s	3.65s	5.24s	7.09s
Traffic sent	51.5KB	243.4KB	605KB	1.2MB
Traffic sent per node	5.1KB	12.2KB	20.1KB	30.9KB
   +------------------------+----------+----------+----------+----------+

                      Table 3: String topology (1)

   +------------------------+----------+----------+----------+----------+
   | Number of nodes | 50 nodes | 60 nodes | 70 nodes | 80 nodes |
   +------------------------+----------+----------+----------+----------+
Convergence time	8.79s	11.11s	12.87s	15.03s
Traffic sent	2MB	3MB	4.1MB	5.6MB
Traffic sent per node	40.5KB	50.4KB	59.2KB	70.1KB
   +------------------------+----------+----------+----------+----------+

                      Table 4: String topology (2)

   These results show that in a linear network, convergence time is
   linear in the number of chained nodes, but the overall transmitted
   traffic is not. This can be easily explained as the convergence time
   reflects the time for both extremities to discover each other
   (updates have to traverse the whole string), but in the meantime,
   other updates are sent. The slope of the convergence time line is
   0.18 second per node, which is pretty quick, but also comes at the
   cost of transmitting multiple updates before all nodes are
   discovered.

4.3. Results for the full mesh topology

```
+----------------------+----------+----------+----------+----------+
|    Number of nodes   | 10 nodes | 20 nodes | 30 nodes | 40 nodes |
+----------------------+----------+----------+----------+----------+
|   Convergence time   |   1.71s  |   3.2s   |   4.83s  |  *6.19s  |
|         ----         |   ----   |   ----   |   ----   |   ----   |
|     Traffic sent     | 202.7KB  |   1.6MB  |   6.6MB  |  18.1MB  |
|         ----         |   ----   |   ----   |   ----   |   ----   |
| Traffic sent per node|  20.3KB  |  83.5KB  | 222.1KB  | 453.8KB  |
+----------------------+----------+----------+----------+----------+
```

   *: the average value was calculated over the results of 9 experiments
          because one of the value was significantly greater.

                     Table 5: Full mesh topology (1)

```
+----------------------+----------+----------+----------+----------+
|    Number of nodes   | 50 nodes | 60 nodes | 70 nodes | 80 nodes |
+----------------------+----------+----------+----------+----------+
|   Convergence time   |  10.64s  |  13.02s  |  15.33s  |  17.93s  |
|         ----         |   ----   |   ----   |   ----   |   ----   |
|     Traffic sent     |  49.1MB  |  95.8MB  | 167.4MB  | 271.9MB  |
|         ----         |   ----   |   ----   |   ----   |   ----   |
| Traffic sent per node| 983.1KB  |   1.6MB  |   2.4MB  |   3.4MB  |
+----------------------+----------+----------+----------+----------+
```

                     Table 6: Full mesh topology (2)

   An incident similar to the one that occurred with a single link
   topology was observed. Although the convergence time is low, the
   amount of transmitted traffic is very high compared to other
   topologies. It can be explained by the high number of distinct links
   (equal to N(N-1)/2): Trickle has to converge on each individual link.
   Unlike the single link topology, this situation is harder to detect
   and resolve.

4.4. Results for the tree topology

```
+-----------------------+----------+----------+----------+----------+
|    Number of nodes    | 10 nodes | 20 nodes | 30 nodes | 40 nodes |
+-----------------------+----------+----------+----------+----------+
|    Convergence time   |  1.16s   |  1.57s   |  1.86s   |    2s    |
|         ----          |   ----   |   ----   |   ----   |   ----   |
|      Traffic sent     |  40.7KB  | 166.7KB  |  374KB   | 644.5KB  |
|         ----          |   ----   |   ----   |   ----   |   ----   |
| Traffic sent per node |  4.1KB   |  8.3KB   |  12.4KB  |  16.1KB  |
+-----------------------+----------+----------+----------+----------+
```

                     Table 7: Tree topology (1)

```
+-----------------------+----------+----------+----------+----------+
|    Number of nodes    | 50 nodes | 60 nodes | 70 nodes | 80 nodes |
+-----------------------+----------+----------+----------+----------+
|    Convergence time   |  2.33s   |  2.42s   |  2.56s   |   2.6s   |
|         ----          |   ----   |   ----   |   ----   |   ----   |
|      Traffic sent     |   1MB    |  1.3MB   |  1.9MB   |  2.4MB   |
|         ----          |   ----   |   ----   |   ----   |   ----   |
| Traffic sent per node |  20.2KB  |  22.8KB  |  26.7KB  |  29.9KB  |
+-----------------------+----------+----------+----------+----------+
```

                     Table 8: Tree topology (2)

   As expected in a tree topology, the convergence time is sub-linear.
   We also oberve that the overall amount of traffic (and per node) is
   relatively low compared to other topologies. This is quite
   satisfying as the tree and double-tree topoligies are the more
   realistic ones.

4.5. Results for the double-tree topology

Number of nodes	10 nodes	20 nodes	30 nodes	40 nodes
Convergence time	1.04s	1.44s	1.5s	1.7s
----	----	----	----	----
Traffic sent	66.9KB	265KB	605.1KB	1MB
----	----	----	----	----
Traffic sent per node	6.7KB	13.2KB	20.2KB	25.3KB

                 Table 9: Double-tree topology (1)

Number of nodes	50 nodes	60 nodes	70 nodes	80 nodes
Convergence time	1.96s	1.98s	2.06s	2.09s
----	----	----	----	----
Traffic sent	1.5MB	2MB	2.8MB	3.5MB
----	----	----	----	----
Traffic sent per node	30.8KB	33.2KB	39.7KB	44.7KB

                 Table 10: Double-tree topology (2)

   Results are quite similar to the simple tree topology. The
   convergence delay is very satisfying while the overall amount of
   traffic is pretty low compared to other topologies.

5. Conclusion

   DNCP does converge in small networks as well as larger ones. It
   converges fast at the cost of a quite high overall transmitted amount
   of data. It behaves particularly well in tree topologies as well as
   double tree topologies, which are the most realistic tested
   topologies.

   The first observed concern appears in the single link topology, where
   the amount of traffic grows dramatically with the number of nodes.
   The problem is known and DNCP actually proposes a solution to that
   problem. But this solution is optional and was not part of the
   tested implementation. Further attention may be necessary on these
   particular mechanisms in order to make sure that DNCP behaves well in
   such situations (if such topology is in scope of DNCP at all).

   The overall amount of traffic was also significant in full mesh
   topologies as well as string topologies. A possible approach to
   improve these results might be to rate-limit the amount of updates

that may be made in short periods of time. Such approach would
provide fast convergence after small changes and would reduce the
overall amount of traffic in reaction to dramatic changes.

Once again, it is important to note that DNCP is expected to be
verbose in case of regular or dramatic changes. DNCP users should
make sure that the network eventually stabilizes, such that DNCP can
take full advantage of the Trickle algorithm.

Next iterations of this document might include further results such
as:

o Measures with libdncp2 instead of libdncp(v1).

o Measures with different profiles (rather than the HNCP profile
   alone).

o Other metrics (e.g., convergence ratio).

o Other scenarios (e.g., no updates at all, sporadic updates).

6. Security Considerations

This document does not specify a protocol or a procedure. DNCP's
security architecture is described in Section 8 of
[I-D.ietf-homenet-dncp].

7. IANA Considerations

This document contains no action for IANA.

8. Informative References

[I-D.ietf-homenet-dncp]
          Stenberg, M. and S. Barth, "Distributed Node Consensus
          Protocol", draft-ietf-homenet-dncp-07 (work in progress),
          July 2015.

[I-D.ietf-homenet-hncp]
          Stenberg, M., Barth, S., and P. Pfister, "Home Networking
          Control Protocol", draft-ietf-homenet-hncp-07 (work in
          progress), July 2015.

[I-D.ietf-homenet-prefix-assignment]
          Pfister, P., Paterson, B., and J. Arkko, "Distributed

                Prefix Assignment Algorithm",
                draft-ietf-homenet-prefix-assignment-07 (work in
                progress), June 2015.

    [RFC6206]   Levis, P., Clausen, T., Hui, J., Gnawali, O., and J. Ko,
                "The Trickle Algorithm", RFC 6206, March 2011.

    [1]   <https://github.com/sbyx/hnetd/>

    [2]   <https://github.com/jech/shncpd>

Appendix A. Acknowledgments

   The authors would like to thank Markus Stenberg for his help with
   hnetd and the remarkable quality of his code, as well as Juliusz
   Chroboczek for the new (yet untested) DNCP implementation.

Authors' Addresses

   Kaiwen Jin
   Ecole Polytechnique / Cisco Systems
   Palaiseau,
   France

   Email: kaiwen.jin@master.polytechnique.org

   Pierre Pfister
   Cisco Systems
   Paris,
   France

   Email: pierre.pfister@darou.fr

   Jiazi Yi
   LIX, Ecole Polytechnique
   Route de Saclay
   Palaiseau 91128,
   France

   Phone: +33 1 77 57 80 85
   Email: jiazi@jiaziyi.com
   URI: http://www.jiaziyi.com/