Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

State Decorrelation Analysis in Information-Centric Networking

a simulation approach

Markus Vahlenkamp

Fakultät Technik und Informatik
Studiendepartment Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Contents

List of Figures

List of Tables

Listings

1 Introduction

The use case of the Internet evolved and is still doing so. It was designed and initially used as a host-centric system. However, nowadays the content-centric use case is shifting into focus with increased momentum. The ratio of content, which is disseminated over and over again, is steadily rising. This is amongst others caused by the widespread popularity of big content platforms like social networks, Video-on-Demand services and the like.
To account for this reinforced content dissemination, techniques like Content Delivery Networks (CDNs) have been invented and deployed in large scale throughout today's Internet. CDN operators make use of the fact that the very same content is repeatedly requested, by utilising caches, request-redirection and the like. CDNs though, are used to cope with the arising increase of required bandwidth or the need to reduce latency. The methods used by CDNs comprise mostly of subsequently applied hacks to the underlying Internet infrastructure like the DNS system or the HTTP protocol.
Information-Centric Networking (ICN) as an actual draft of the next-generation network architecture takes the need for smart content dissemination into account. ICN aims at dragging content awareness into the network and makes it a first class citizen. The network is queried for content via content names, thereupon the network locates, requests and transfers the content towards the requesting party. Throughout this dissemination process, the content is cached on all traversed nodes throughout the network and is held available for consecutive dissemination. All of these mechanisms like caching, location independent content identification, content authentication etc. are considered to be inherent parts of the ICN paradigm. These changes to the basic principals approach different shortcomings of today's networking, but also introduce new challenges.
The control plane of the network is influenced by the data plane. Whenever content is requested or otherwise published, control plane states are created and result in a situation, where regular users of the network interact and directly influence the control plane of network devices. A condition that is unimaginable in today's networks – for good reasons – like we already illustrated in [14, 13].
Within NDN [16], the most popular ICN scheme so far, the control plane maintains Reverse Path Forwarding (RPF) states that are used to forward the requested content towards the subscriber. We will investigate, through the use of a simulation environment, how a decorrelations of these states may manifest itself, as well as its impact on the overall network.
The reminder of this paper is organised as follows. In Section 2 we will give an overview of the problem space and how we want to examine the impact of State-Decorrelation by using a simulation approach. In Section 3 we will first elaborate on the characteristics that need to be kept in mind when using the simulation approach to gather meaningful data for system behaviour evaluation. Following, different Named Data Networking (NDN) / CCNx simulation implementations will be presented with an emphasise on their key features and intended use cases. Section 4 starts with a more comprehensive introduction of the ndnSim

implementation, the one we use for evaluation. Subsequent we continue by presenting our actual simulation environment setup. Results of the conducted simulation runs are illustrated and discussed in Section 5, which is eventually followed by an elaboration of conclusions we can draw from our simulative examination approach.

2 Problem Statement

By introducing a new networking approach ICN tries to solve or mitigate different issues that exist in today's networks. However, by introducing new techniques, also new challenges are likely to arise that need further investigation.
One of these challenges is the stateful forwarding, that is introduced by some of the ICN schemes. This statefulness may lead to State Decorrelation issues, in which different nodes still maintain the state of a particular content transfer, that others already dropped, for whatever reason. Especially the most popular NDN / CCNx implementation is vulnerable to these State Decorrelation issue.
The NDN / CCNx approach introduces new states to the network, namely control plane states, which are driven by the data plane. Each subscription results in a trace of Reverse Path states in CCNx forwarding nodes between subscribers and an actual source of the requested content. Theses soft-states need to be stored, timers need to be maintained and a purging mechanism has to be executed in case of timer expiry or when the subscription is satisfied. If, however, states are dropped, prior to their timer expiry and also before the data arrives, the information where to forward the content to, is lost, and further delivery is interrupted. Due to the fact that each node on the path needs to maintain those RPF states, the entire content dissemination relies on a consistent and reliable end-to-end RPF state chain, which results in a multiplication of possible points of failures. If just one node misses such RPF state, for whatever reason, the content propagation is interrupted. Hence, the states need to be reestablished via reissuing the Interest, an operation, that results in an even increased processing load of the involved nodes. Depending on the reason for the prematurely state drop, such as an overload situation, the increased state maintenance may even aggravate the whole situation.
In our previous work [15, 14, 13], we deployed the NDN prototype implementation CCNx within a testbed environment consisting of five routers, which formed a network of two routes. With its five CCNx nodes the network was quite small. Since ICN is intended to also be implemented in larger networks, like in Internet Service Provider (ISP) or even on Internet scale, the expressiveness of such small setup is limited. To better understand the behaviour and effects, we will evaluate larger topologies throughout this work. Hence, we are going to implement a simulation environment through which we will be able to analyse the interaction of larger amounts of NDN / CCNx content routers.

3 Assessment

A simulation has specific characteristics compared to a testbed. In the following Subsection, we will discuss these special attributes and conduct a critical examination of risks and shortcomings that go with the systems analysis via a simulation approach. In Section 3.2 we will then introduce and compare different NDN simulation implementations that are publicly available and reason on our choice for a simulation environment.

3.1 Simulation Characteristics

Since the implementation in a large scale testbed is out of reach, we opt to implement an extensive examination environment by utilizing simulation tools. Doing so, we have to take the peculiarities and characteristics of simulations compared to the actual code execution into account. In general, simulation approaches are deterministic, as long as there is no randomness explicitly introduced in the input parameters, consecutive runs will always yield exactly the same results. This is also due to the fact that all operations are executed solely within the memory of the simulation host. Even the time within a discrete simulation environment is most likely decoupled from real time. This decoupling of simulation time and real time is also a basic requirement to be able to simulate the operation of a large amount of nodes on a simulation host. Simulations are meant to imitate the operations of an arbitrary number of entities, in our case the CCNx content router nodes. This is sometimes only viable, because the code used within the simulation framework is not the original implementation code. By reducing the functionality and simplifying the system, the computational effort and the complexity is reduced. This reduction and simplification, however, is just applicable up to a point, where the underlying procedures and principals still exist. The outcome of this down-stripping approach is the lowered footprint of the simulation compared to the real implementation, which helps to build a scalable environment for data gathering.
The handling of a simulation in most cases is fairly easy compared to the management of a testbed environment. A scenario description is interpreted and executed by the simulation framework. Predefined events will be processed throughout the simulation. Everything that happens within the simulation is observed in exact causal relations to one another. Hence, it is possible to analyse the accruing events and their related effects throughout the entire network.
Nevertheless, the simulation approach relies on an accurate model creation. The model needs to follow the basic principals of simulation modelling, as defined by [12]. Thus a model needs to fulfill the mapping, reduction and pragmatism criteria. The mapping criteria predicates that an object of the world that is to be modeled, needs to be mapped to an object within the simulation environment. The reduction criteria allows the model to comprise of less properties than exist in the real word. Whereas the pragmatism criteria claims that the

model serves a purpose, hence, the model has to contain all properties necessary to serve its purpose.
These criteria introduce the problem of deciding which properties are necessary and which are not. When leaving out certain parts of the model, it may be the case that the simulation models behaviour diverges from the real implementation behaviour. In our case, this could lead to conclusions that may not apply to the real implementation.
The environment, that consists of a discrete-event simulation, acts deterministically. This holds the advantage that measurement results are well reproducible. On the other hand, many effects in real code execution arise through non-deterministic events, like for instance race-conditions or the like. We are not interested in race-conditions or similar implementation effects, but the resulting overall network behavior. Nonetheless, some effects may not arise within the simulation because of the lack of non-determinism. Further, as described in [9] discrete-event simulations are systems in which events, or state-variable changes just appear at discrete points in time. Thus it is possible to order the occurring events of the simulation in exact chronological order. One way of dividing discrete-event simulations is by classifying them as deterministic or stochastic. Within a deterministic system the same input configuration results in the same simulation output, whereas the stochastic model comprises randomized input into the simulation, which leads to randomised output values. The value range of randomized variables of course has to be constrained to reasonable values, such that the simulation output can be treated as a statistic estimate of the characteristics of the underlying system.

3.2 Framework Comparison

Different projects exist that implement the NDN operations in simulation frameworks. Some of them are proprietary, others rely on well known network simulation frameworks. Hereinafter, we will give a short overview about different NDN / CCNx simulation implementation as well as their features.
Table 1 shows a comparison of the following four simulators: CCNPL-SIM [1] that runs on top of CBCBSim [5], ccnSim [2], an Omnet++ [7] module, DCE [4] and ndnSim [3] which are both NS-3 [6] modules.
While the DCE simulation allows the execution of the real code of the CCNx prototype, the other simulation tools rebuild the behavior of NDN within their environment. Debugging and tracing functionalities are supported by each of the four candidates.
In terms of their scalability, the ccnSim software is performing best, which means that ccnSim has the lowest resource requirements per simulation node. On the contrary DCE is performing worst, which is due to the execution of the real CCNx code. The real code execution contains more overhead then the abstract simulation implementations. Further the deployment status of DCE is rated lower than of the other simulation implementations. Since we aim to simulate large topologies, the DCE implementation seams inadequate due to its lack

	ccnSim	CCNPL-Sim	DCE	ndnSim
Real code execution	✗	✗	✓	✗
Debugger support	✓	✓	✓	✓
Tracing support	✓	✓	✓	✓
Scalability	+++	?	+	++
Deployment	++	++	+	++

Table 1: NDN simulations comparison [10]

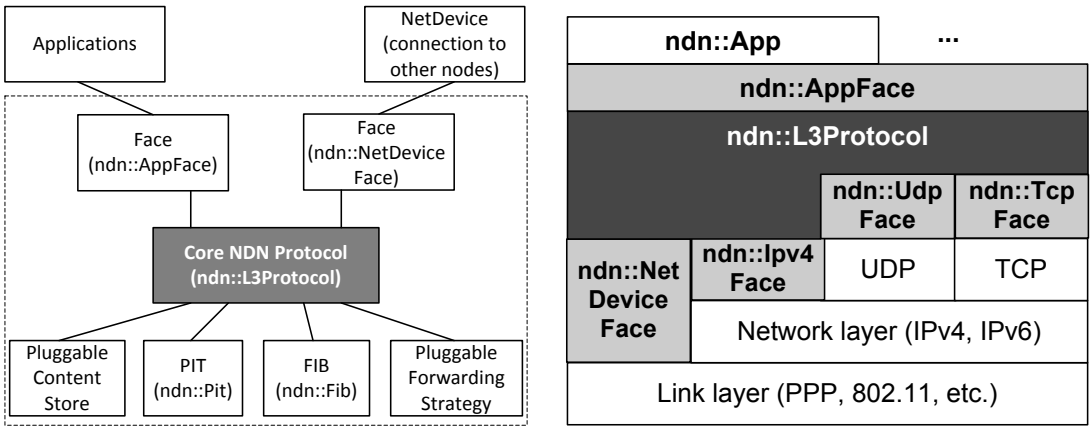
of scalability. The ccnSim implementation is performing well in case of scalability, but primarily focuses on the caching behavior research. Hence, we opt for the ndnSim simulation as the basis for our studies, it is well documented and developed by the University of California, Los Angeles (UCLA), which is also involved in the development of CCNx.
ndnSim is build up on the open source network simulator NS-3 [6], which is a discrete-event network simulator.

4 Implementation

In what follows, we will take a closer look at the ndnSim framework by introducing the components, that the framework consists of. Subsequent in Section 4.2 follows a detailed overview of our actual simulation setup as well as an explanation of the way we processed and analysed the gathered data.

4.1 ndnSim Framework

The ndnSim implementation comprises of the components illustrated in Figure 1(a). The central component of the protocol implementation is the *ndn::L3Protocol*. It implements the core NDN functionality of receiving and processing Interests, as well as the handling of incoming data packets. Above the *ndn::L3Protocol* two abstract Faces are available. The *ndn::AppFace* that serves as an interface towards locally executed applications, and the *ndn::NetDeviceFace* that acts as an abstraction of a network interface connecting the local with other content routers. The *ndn::L3Protocol* itself uses different interfaces to connect to other ndnSim simulation nodes. The *ndn::NetDeviceFace* operates directly on top of the link layer, which may be an abstract PPP, 802.11, etc. implementation. There also exist other interfaces, like *ndn::Ipv4Face*, *ndn::UDPFace* and *ndn::TCPFace*, which build up on IPv4 on the network layer, and respectively TCP or UDP on the transport layer.
The *ndn::L3Protocol* internally relies on the Pending Interest Table (PIT) and the Forwarding Information Base (FIB) as shown in Figure 1(a). The PIT is keeping record of active Interests that have already been forwarded, but the requested data has not yet arrived. The FIB keeps



(a) Components (b) Communication layers

Figure 1: ndnSim overview [8]

track of the routing information, where to send Interests that the router is not able to satisfy via its cache. It manages namespace to Face associations. Further also Content Store and Forwarding Strategy plugable modules exist. By default a simulated content router is not caching any content locally. The Content Store interface offers these capabilities. Three different Content Store implementations, which differ in their cache replacement policy, are also available.

The *ndn::ForwardingStrategy* abstraction is used to handle core functionalities for Interest and data forwarding. Figure 2 depicts the hierarchy of different Forwarding Strategies. Just *Flooding*, *BestRoute* and *SmartFlooding* are full implementations, whereas *Nacks* and *GreenYellowRed* are abstract extensions. The *Nacks* abstraction allows for negative acknowledge of Interests, for CCNx nodes to indicate that the Interest can not be forwarded any further. To use this functionality of negative acknowledgement though, it has to be explicitly enabled, otherwise no NACKs are generated. The *GreenYellowRed* forwarding strategy classifies Faces as green, yellow and red, according to their status. Faces are being marked as Green, when the Face works correct, which indicates that data is returned for Interests that are delivered to that particular Face. The yellow marking indicates that the status of the Face is unknown, this may be the case, when the Face was just recently added or was not used for some time. Faces marked red are those that do not perform forwarding as expected and thus should not be used for Interest forwarding. When utilizing the *Flooding* implementation, which is a direct inheritance of *Nacks*, incoming Interests are flooded out of all Faces listed in the FIB, except the incoming Face. The *BestRoute* implementation makes use of the Face classification and uses the highest ranked matching green or yellow route to forward an Interest. The *SmartFlooding* implementation also utilizes the highest rated green Face, but in contrast to the *BestRoute* implementation the Interests are flooded towards all yellow Face

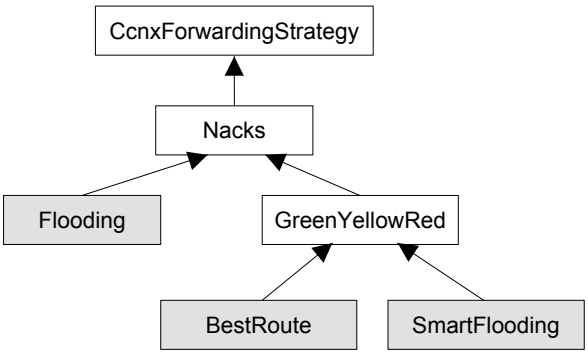


Figure 2: ndnSim forwarding strategies [8]

if no green Faces are available. Red Faces, however, are not used for Interest forwarding by any of the two Forwarding Strategies.

4.2 Simulation Setup

We utilize the NS-3 based network simulator, ndnSIM [8], in the version as of the 6th November 2012, to extend our analysis of the impact of data-driven states on ICN. Further we make use of the Sprintlink topology #1239, provider by the Rocketfuel [11] topology mapping engine with its 315 nodes to form our simulation core topology. These core nodes are interconnected by point-to-point links with a bandwidth of 10 Mbit/s and the corresponding latency values derived from the topology file. The topology is further extended by three additional edge nodes that are created per each core router. The connections between each core router and its associated edge nodes is established via links of 1 Mbit/s with a fixed latency of 10 ms. Figure 3 illustrates the resulting topology via a screenshot of the simulator gui.
Since we want to study the accruing effects of data-driven states, along with the impact of State-Decorrelation, the nominal bandwidth of the links carries no meaning, hence for the sake of simplified simulation conduction we stick to these low bandwidth values.
Every simulation node is provided with a protocol stack consisting of the link-layer Face *(ndn::NetDeviceFace)* and the NDN protocol *(ndn::L3Protocol)* implementation. For the ForwardingStrategy, the module that defines how Interests and data are being forwarded, the *ndn::BestRoute* implementation is used, whereas the ContentStore module is not used, and hence is left uninstantiated in our configuration. It is also worth mentioning that the maximum size of the Pending Interest Table of each node is not explicitly limited.
To generate traffic in the network, we utilize the *ndn::ConsumerCbr* and *ndn::Producer* applications. The Consumer applications issue Interests at a configurable frequency and thus initiate the data transfers. The Producer applications are configured to reply with a data

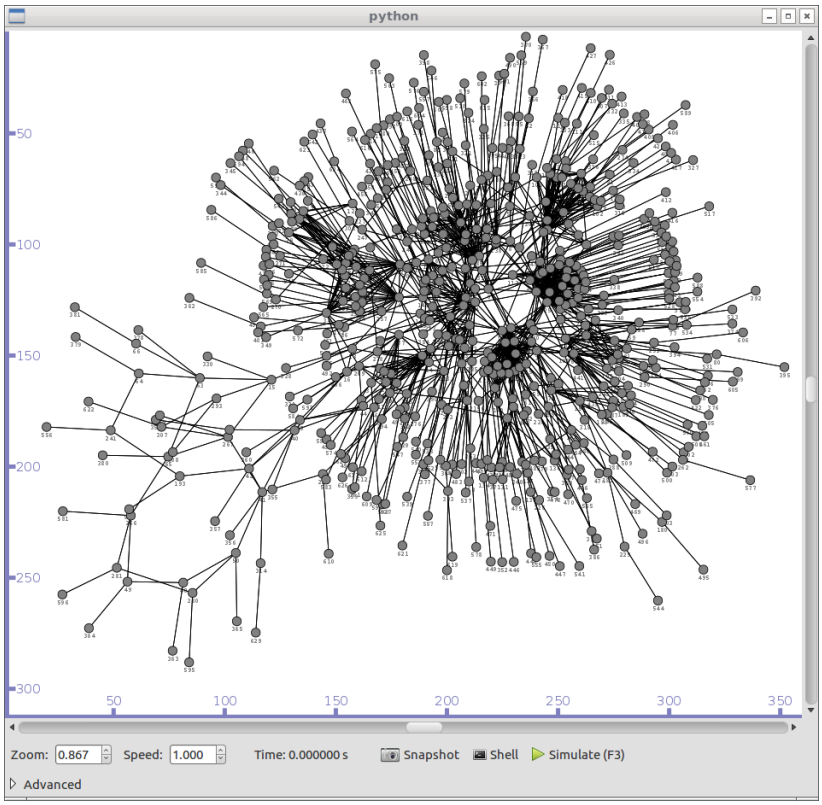


Figure 3: Simulation topology

packet of 1024 Byte in response to each received Interest, which is addressed to their specific namespace.

4.3 Simulation Parametrization

In each simulation run, we create a configurable amount of Producers that are randomly distributed among the network nodes. This placement is, however, constrained to either just core or edge nodes. Regardless of the position, the maximum amount of Producers per node is limited to one. Consumer nodes on the contrary are placed solely on edge nodes and allow for multiple Consumers on the same node. In the case of multiple Consumers per node, it is just assured that different Consumer applications on one node do not issue Interests that are processed by one and the same Producer.
The routing information required to forward Interests towards the content producers is calculated and provided by a helper class that is shipped with the ndnSim simulator. The helper class is aware of the available content providers, the namespace, they provide data for, just as the overall network topology. This way the routing information is pre-computed by the

ndn::GlobalRoutingHelper and the content routers are automatically fed with the static routing information in the simulation initialisation phase.
To give credit to different constellations that may occur in the network, the amount of producers to be scattered throughout the network is adjustable, the same applies to the consumers, which are configured as amount of consumers per producer. Since the *ndn::ConsumerCbr* implementation is used, the frequency of interest issuance is adjustable. We further distinguished the positioning of producer applications by attaching them just to core or just to edge nodes. This is due to the bandwidth available in core or at the edge. The bandwidth for core and for edge links respectively is also globally adjustable. The latency between core nodes as already mentioned is taken from the topology file, whereas the latency between core nodes and their attached edge nodes is configurable for all of these links at once. At last the runtime of each simulation scenario is pre-definable, such that a batch of given simulation runs can be executed.
To analyse the network behaviour we need to gather different kinds of data. All of the following measurement results are aggregated on a per second accuracy and will subsequently be persisted for later evaluation. The data is subdivided into four categories: the per chunk transmission times, the amount of transmitted data, interest retransmissions and the PIT stats.
The chunk transmission time category covers the amount as well as the minimal, maximal and mean times, of all of the chunks that successfully reached the Consumer that requested them within each particular second. The amount of transmitted data is the value that mirrors the amount of data chunks that got completely and successfully delivered to the Consumers. The category of interest retransmission covers the sum as well as minimal, maximal and mean amount of Interest retransmission issued by the aggregate of all Consumer nodes. The same applies for the PIT states.
Because of the random scattering of Consumers and Producers within the network, traffic flows will cross each other and also share various links. Hence, the maximal available bandwidth the network is able to provide is reduced and does not equal the sum of Consumer interface bandwidths. To figure out the bandwidth that the network is able to provide in a specific constellation, we make use of the algorithm illustrated in listing 1.
The results of these calculations provide a rough estimate of the Bandwidth, which the Network is able to provide in direction of the content transfers and can in conjunction with the transfer statistics, further be used to evaluate the content transfer efficiency of the network and thus provide indications of the impact of decorrelating states.

```
# Links are directed, bidirectional connection between
# two nodes are for this algorithm seen as two links

Calculate shortest path for each Producer towards its Consumers

For all links on each path do
   take not yet marked, link with smallest per flow capacity
   mark all not yet marked flows through that link with available per ↘
      →flow bandwidth
      available per flow bandwidth = (link bandwidth - bandwidth sum of↘
         → already market flows) / (remaining unmarked flows)
done

Overall network capacity = sum bandwidth of each flow
```

Listing 1: Network bandwidth capacity calculation algorithm

5 Results

We utilise the setup described in Section 4.2 to run different simulation scenarios. For each scenario depicted here, we create 20 Producers, which are randomly scattered, either just in the networks core or solely at the networks edge.
Figure 4 illustrates the results of a simulation run with 20 Producer nodes placed at the network core and 100 Consumers at the networks edge, which are requesting content chunks. With a frequency, that fully utilizes there 1 Mbit/s link to the network. The maximum transmission times for chunks reside on a low non-increasing level. The maximum retransmissions fluctuate slightly above 100 retransmissions per second. The max. Pending Interest count stagnates at a level of ≈ 350 PI's. The calculated overall network bandwidth between the Consumers and their Producers of 99.0 Mbit/s is slightly above the 'completed data transfers' value of ≈ 89 Mbit/s, which is because just the explicit content data that is received is included in this graph, interest and content packet header data is left out. Figure 5 further illustrates information regarding the paths from Producers towards Consumers. 98 of the paths between Producer and Consumer provide a Bandwidth of approximately 1 Mbit/s, whereas 2 paths just provide a capacity of 0.5 Mbit/s. The right hand diagram gives an impression on the length of paths that exist between Producers and Consumers. Most of the paths consist of 6 hops, whereas the longest paths are 11 and the shortest paths are 2 hops long. The whole network in this case is balanced, there is no harmful increase in transmission times, PIT entries are stable and retransmits are also on an acceptable level.
Figure 6 now shows a different picture. Looking at 200 Consumers, the total amount of Consumer nodes has doubled, while the amount of 20 Producers remains unchanged. The maximum transmission times begin to rise with local maxima that grow linear. Even at the

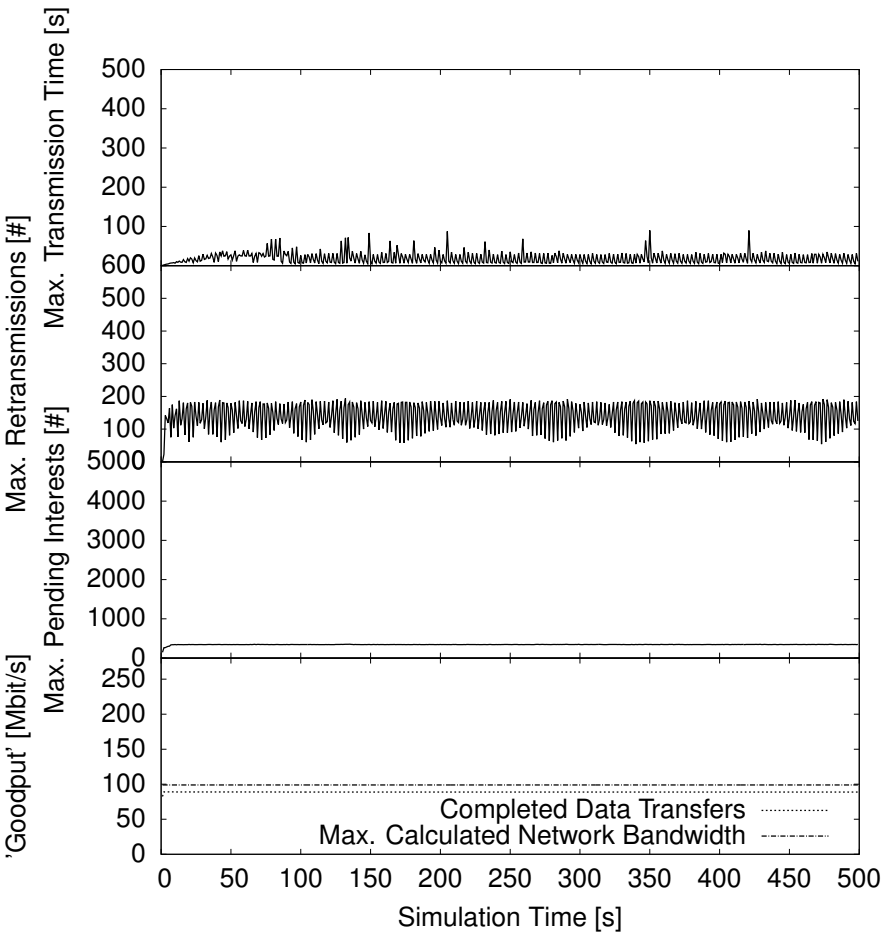


Figure 4: Simulation run; 20 Producer located at core nodes with 100 Consumers nodes at the network edge

end of the simulation run content arrives that was initially requested at the beginning of the transmission. The maximum Retransmissions also rise up to 1000 Retransmits per second. This is due to the increased Transmission Times in conjunction with the softs-states that need to be refreshed before the associated timer expires. Also the Pending Interest count increases as a result. The calculated maximum end-2-end network content capacity also increased by 85 Mbit/s up to 184 Mbit/s. The reason for this sublinear increase is visible when taking figure 7 into account. Now there exist even more paths, which do not allow their attached Consumer nodes to utilize their full 1 Mbit/s uplink capacity. What remains is the Gaussian like distribution of link-lengths with its peek at 5 hops per link.
Compared to figure 6, figure 8 shows an even more stabil and hence severe increase in maximum Transmission Times. The tendency of a 45° slope is easily noticeable. Even in the end of the simulation, packets that have first been requested at the start of the simulation con-

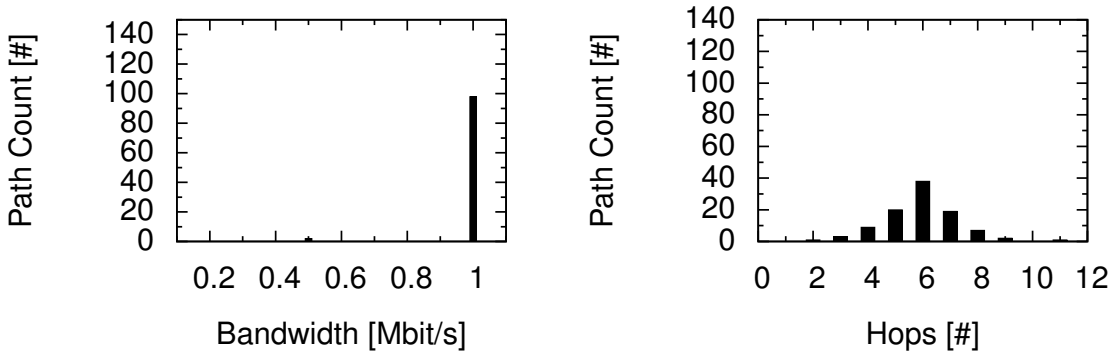


Figure 5: Calculated path bandwidth and hop count overview (related to Figure 4)

tinuously arrive. However, the maximum Retransmissions on average just increase slightly, from ≈ 400 to ≈ 440. The max. Pending Interests on the contrary increased by a factor of 5 up to slightly below 5000 PIT's. The maximum Calculated Network Bandwidth rose up to ≈ 250 Mbit/s. As visible in figure 9 the available per path bandwidth declined further, such that the average bandwidth is in this constellation 0.546 Mbit/s.

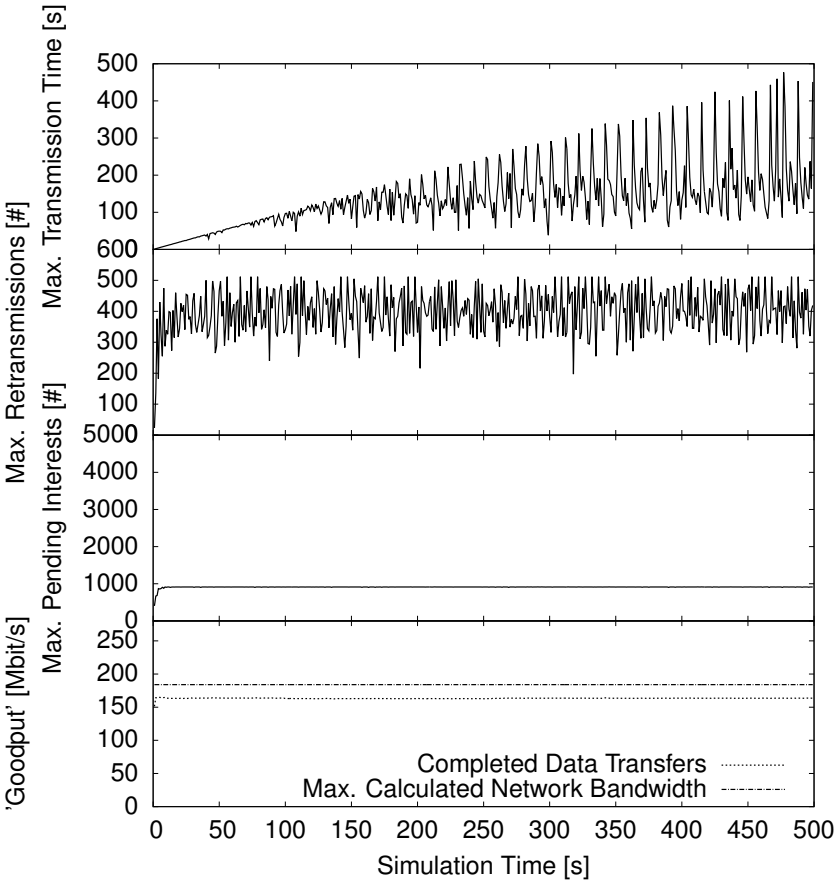


Figure 6: Simulation run; 20 Producer located at core nodes with 200 Consumers at the network edge

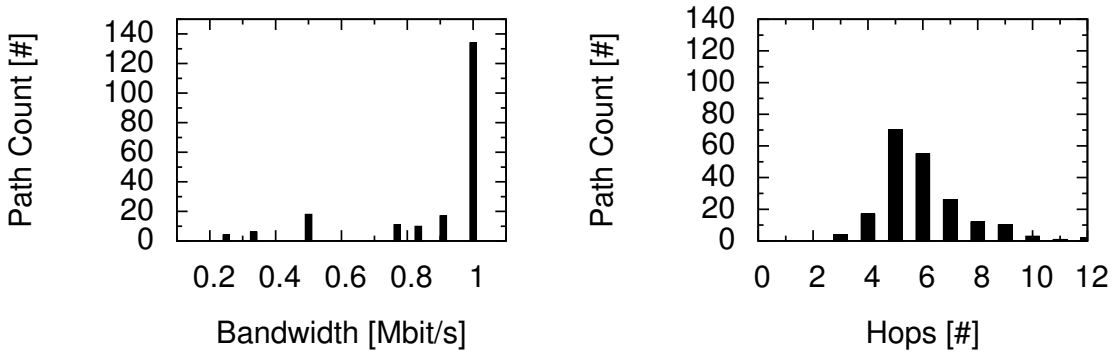


Figure 7: Calculated path bandwidth and hop count overview (related to Figure 6)

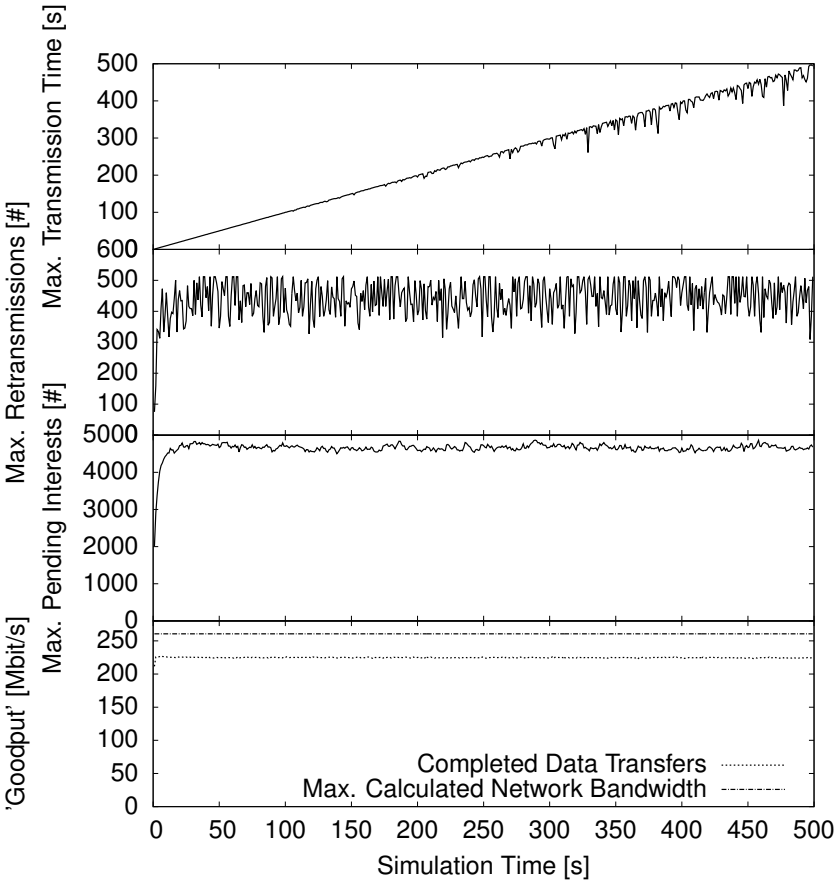


Figure 8: Simulation run; 20 Producer located at core nodes with 400 Consumers at the network edge

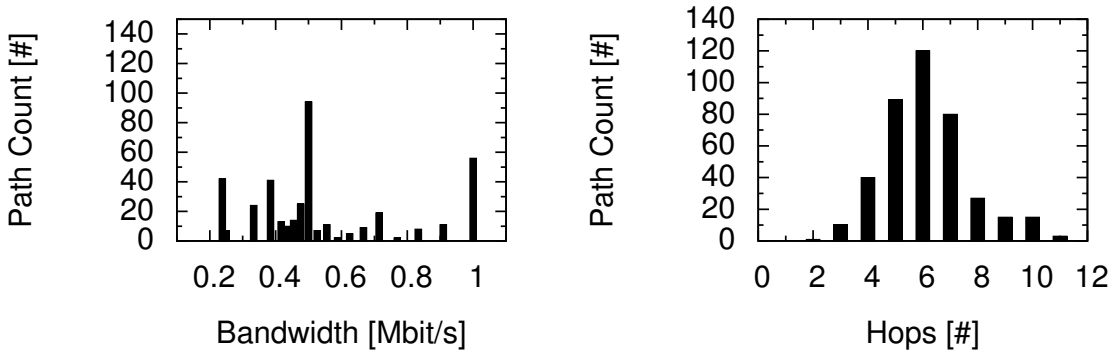


Figure 9: Calculated path bandwidth and hop count overview (related to Figure 8)

6 Conclusion & Outlook

Since the simulation is a discrete-event simulation, in which calculation events do not consume time, and processing results are instantly available, measured in simulation time, all the nodes within our simulation environment behave like optimal nodes. They do not experience resource overload situations like nodes of our testbed.
In the testbed, the overall performance was influenced by the maintenance of states that each node had to fulfill. Within the simulation environment the processing power of each node does not influence the simulation due to its timewise decoupling. Their processing load and complexity is hidden within the events, which appear at a certain point in time. The occurring events are processed at the very point in time they arise and the result is, measured in simulation time, instantly available. Bandwidth consuming events form the only exception, for them the transmission time is explicitly take into account. Due to these basic conditions the system performance in terms of network goodput is regardless of the amount of consumers and thus the amount of Pending Interests within the network residing at a level above 85%. To be precise 86% in Figure 4, 88% in Figure 6 and 89% in Figure 8. This does not allow the conclusion of any kind of systematic issue.
Nevertheless, like already questioned and discussed in Section 3.1, we must acknowledge that the ndnSim implementation like we utilized it throughout this work, does not fit the requirements concerning the right level of abstraction that is needed for the evaluation of the State Decorrelation case in CCNx. Nevertheless do we plan to run these simulations once again, in a slightly different setup. Next we will use the random and persistent PIT policies, to limit the available resources in a way, that they are also limited in a real deployment scenario. By doing so, we anticipate to be able to gather results that clearly illustrate the impact of State Decorrelation.

References

[1] “The CCNPL-SIM Homepage,” http://code.google.com/p/ccnpl-sim, 2012.

[2] “The ccnSim Homepage,” http://perso.telecom-paristech.fr/~drossi/index.php?n=Software.ccnSim, 2012.

[3] “The ndnSim Homepage,” http://ndnsim.net/, 2012.

[4] “The NS3 DCE CCNx Homepage,” http://www-sop.inria.fr/members/Frederic.Urbani/ns3dceccnx/, 2012.

[5] “The CBCBSim Homepage,” http://www.inf.usi.ch/carzaniga/cbn/routing/, 2013.

[6] “The NS-3 Homepage,” http://www.nsnam.org/, 2013.

[7] “The Omnet++ Homepage,” http://www.omnetpp.org/, 2013.

[8] A. Afanasyev, I. Moiseenko, and L. Zhang, “ndnSIM: NDN simulator for NS-3,” NDN, Technical Report NDN-0005, October 2012. [Online]. Available: http://www.named-data.net/techreport/TR005-ndnsim.pdf

[9] J. Banks and J. Carson, *Discrete-event system simulation*, ser. Prentice-Hall international series in industrial and systems engineering. Prentice-Hall, 1984. [Online]. Available: http://books.google.de/books?id=wWFRAAAAMAAJ

[10] D. Camara, F. Urbani, M. Lacage, T. Turletti, and W. Dabbous, “Experimentation with ccn,” INRIA, Planete-Project, Presentation, 2012. [Online]. Available: http://www.ccnx.org/wp-content/uploads/2012/08/1Lacage.pdf

[11] R. Mahajan, N. Spring, D. Wetherall, and T. Anderson, “Inferring link weights using end-to-end measurements,” in *Proc. of the 2nd ACM SIGCOMM Workshop on Internet measurment (IMW’02)*, ser. IMW ’02. ACM, 2002, pp. 231–236.

[12] H. Stachowiak, *Allgemeine Modelltheorie*. Wien, New York: Springer-Verlag, 1973. [Online]. Available: http://books.google.de/books?id=DK-EAAAAIAAJ

[13] M. Wählisch, T. C. Schmidt, and M. Vahlenkamp, “Backscatter from the Data Plane — Threats to Stability and Security in Information-Centric Networking,” Open Archive: arXiv.org, Technical Report arXivarXiv:1205.4778, 2012. [Online]. Available: http://arxiv.org/abs/1205.4778

[14] ——, "Bulk of Interest: Performance Measurement of Content-Centric Routing," in *Proc. of ACM SIGCOMM, Poster Session*. New York: ACM, August 2012, pp. 99–100. [Online]. Available: http://conferences.sigcomm.org/sigcomm/2012/paper/sigcomm/p99.pdf

[15] ——, "Lessons from the Past: Why Data-driven States Harm Future Information-Centric Networking," in *Proc. of IFIP Networking*. Piscataway, NJ, USA: IEEE Press, 2013, accepted for publication.

[16] L. Zhang, D. Estrin, J. Burke, V. Jacobson, and J. D. Thornton, "Named Data Networking (NDN) Project," NDN, Tech.report ndn-0001, 2010.