Distributed multilevel storage infrastructure for visual surveillance in intelligent transportation networks

Chih, Wei Ling

2018

Chih, W. L. (2018). Distributed multilevel storage infrastructure for visual surveillance in intelligent transportation networks. Master's thesis, Nanyang Technological University, Singapore.

<http://hdl.handle.net/10356/74822>

<https://doi.org/10.32657/10356/74822>

**NANYANG
TECHNOLOGICAL
UNIVERSITY**

**DISTRIBUTED MULTILEVEL STORAGE
INFRASTRUCTURE FOR VISUAL
SURVEILLANCE IN INTELLIGENT
TRANSPORTATION NETWORKS**

**CHIH WEI LING
SCHOOL OF COMPUTER SCIENCE AND ENGINEERING
2017**

DISTRIBUTED MULTILEVEL STORAGE INFRASTRUCTURE FOR VISUAL SURVEILLANCE IN INTELLIGENT TRANSPORTATION NETWORKS

CHIH WEI LING

School of Computer Science and Engineering

A thesis submitted to the Nanyang Technological University
in partial fulfilment of the requirements for the degree of
Master of Engineering

2017

Acknowledgements

I am grateful to my supervisor, Prof. Anwitaman Datta, and my co-supervisor, Dr. Jun Xu, whose expertise, understanding, generous guidance and patient support made it possible for me to work on a topic that was of great interest to me.

I would like to express my gratitude to Economic Development Board (EDB) and Western Digital (WD) for providing the excellent opportunity and funding under Industrial Postgraduate Programme (IPP) to work across both academic and industrial environments. I have gained a lot of working experiences and lessons during the process.

Contents

Acknowledgements	i
Contents	iii
Abstract	vii
List of Tables	viii
List of Figures	xi
1 Introduction	1
2 Background Technologies and Concepts	8
2.1 Distributed System in ITS and Cloud Computing	9
2.2 Fog Computing and Software-defined Concepts	10
2.3 Intelligent Visual Surveillance System	11
2.3.1 Evolution of Surveillance Systems	12
2.4 Network Coding Basics	13
2.4.1 Network Coding Model for Single Source Multicasting	14
2.4.2 Local and Global Encoding Vectors	15
2.4.3 Main Theorem of Network Multicast	16
2.5 Software Tools	16
2.5.1 ns-3	16
2.5.2 BRITE	17
2.5.3 Kodo	17
2.5.4 MPEG-4 and Video Trace File	18

3	Related Works and Our Contributions	19
3.1	Related Works	19
3.1.1	Network Coding	19
3.1.2	Network Coding for Multimedia, Video-on-Demand and Live Streaming	21
3.1.3	Network Coding for Networking	25
3.1.4	Simulation of Network Coding	25
3.2	Our Contributions	26
4	A Distributed Multilevel Storage Infrastructure (DMSI) for Visual Surveillance in ITS	31
4.1	Design Goals	31
4.2	Underlying Functionality Requirements	34
4.2.1	Storage and Coding Scheme	34
4.2.2	Network Bandwidth Allocation and Content Placement	36
4.2.3	Fine-grained Control for Reliability	37
5	Network Coding Models for ITS	40
5.1	System Model and Problem Formulation	40
5.2	Linear-Information-Flows Algorithm	44
5.3	Claims and Proofs	47
5.4	Hybrid Network Coding	51
6	Simulation and Results	54
6.1	Simulation Model and Methodology	54
6.1.1	Network Model for Simulation	55
6.1.1.1	Network Topology	55
6.1.1.2	Software-defined Network	57
6.1.1.3	Delay Analysis	58
6.1.2	Network Coding Model and Delivery Scheme	59
6.1.2.1	Video Trace File	59
6.1.2.2	Transmission Policy	59
6.1.2.3	Network Coding Module	60
6.1.3	Simulation Architecture	62

6.1.4	Simulation of Hybrid Network Coding	63
6.1.4.1	Network model	63
6.1.4.2	Network coding model and delivery scheme	64
6.2	Results	65
6.2.1	Comparison between Network Coding Paradigms	66
6.2.1.1	Network Throughput	66
6.2.1.2	Accumulated Network Traffic between Network Cod- ing Paradigms	68
6.2.1.3	Coefficient Tag Overhead	69
6.2.2	Comparison of Hybrid Network Coding against Traditional Network Coding Paradigms	71
6.2.2.1	Network Throughput under Network Change and Churn	71
6.2.2.2	Our Proposal under Other Network Setting	80
6.2.2.3	Network Throughput when No Packet Loss and 0.05 Packet Loss Rate	84
6.2.2.4	Network Throughput under Different Packet Loss Rates	87
6.2.2.5	Further Discussion on Throughput of Hybrid Net- work Coding w.r.t. Parameter Space	92
6.2.3	Summary	93
7	Conclusion and Future Works	94
7.1	Future Works	95
7.1.1	Network Simulation	95
7.1.2	Theoretical Analysis	96
7.1.3	Security	97
	Bibliography	98
A	My Curriculum Vitae	108
A.1	Education	108
A.2	Working Experience	108
A.3	Computer Skills	109

A.4 Publications	109
----------------------------	-----

Abstract

Large volume of data is generated by traffic surveillance devices such as cameras and sensors integrated in an intelligent transportation system (ITS). To deal with the extreme volume and the massively geographically distributed sources of data, we advocate a tiered storage and processing architecture, using edge nodes to augment a centralized backbone, facilitated by network coding and emerging fog computing solutions. For that reason, we design a linear optimization cost model (LO) to capture the network usage (e.g., network bandwidth and storage) with the aid of storage helpers for the network stakeholders. By assuming that the network is static over a window of time, we design an algorithm to construct a deterministic network code that sets up the multicast connections from the source processes to destinations (either storage helpers or data centers) for every feasible solution satisfying the set of graph-theoretic-constraints in the LO model to deal with the networks at small to medium scales. The LO model and the network coding algorithm will be implemented by leveraging OpenFlow, which is a realization of SDN controller. An SDN controller has the global view of the network and operates as a network operating system (e.g., NOX), and thus it is suitable to become the entity to overhaul and make the global decision for ITS. We also emphasize on a hybrid approach of both random and deterministic network coding paradigms to deal with the networks at medium to large scales. We have run several simulations to compare the performance of the ingredients of the hybrid approach by using ns-3 to reconfirm that from the aspects of network throughput and network usage, deterministic network coding performs better than random network coding. We have also compared the performance of the hybrid approach against both network coding paradigms. Under certain network conditions, our simulation results show that the proposed hybrid approach performs better than other approaches.

List of Tables

1	Table of Notations for the System Model	viii
2	Table of Notations for the Algorithm	x
6.1	Parameters for the Scale-free Networks Generated by BRITE	56
6.2	Flat Topology (AS Only or Router Only) Parameters	56
6.3	Top-down Hierarchical topology Parameters	56
6.4	Parameters for the Component Cluster of 150 Nodes Generated by BRITE	64
6.5	Parameters for the Component Cluster of 225 Nodes Generated by BRITE	64

We summarize the set of notations used in this report in the following two tables:

Table 1: Table of Notations for the System Model

Symbol	Definition
$G = (V, E)$	A finite directed acyclic graph with a collection of vertices V and a collection of edges E
$C = \{c_{ij} : (i, j) \in E\}$	The set of edge capacities of G
S, s	The set of sources ($S \subset V, S \cap T = \emptyset$) in the graph G to model smart cameras, a source
T, t	The set of receivers ($S \subset V, S \cap T = \emptyset$) in the graph G to model data centers, a receiver
H, h	The set of helpers ($h \in V \setminus S \cup T$), a helper (intermediate node with storage capacity)
U_h	The available storage bandwidth (download bandwidth) of some helper h
U	The expected total amount of storage able to be processed

	over the next τ time unit (so per unit time is $\frac{U}{\tau}$)
X_{m_s}, R_{m_s}	The source process generated at some source s , the rate of the process
m_s	For convenience, we use the index of the process to denote the process itself, e.g., m_s represents X_{m_s} where $s \in S$.
d	A destination (either a helper or a receiver)
$D, D(d)$	The collection of all source processes, the collection of source processes requested by a destination d
Θ	The set of source processes that are not requested by any receiver $t \in T$
$T(i)$	The collection of receivers that are accessible from some node i
$\Gamma(i)$	A set partition of $D \setminus \Theta$ that represents the sources that can be mixed on links going into node i with receivers $t \in T$
$H(i)$	The collection of helpers that are accessible from some node i
$\Psi(i)$	A set partition of Θ that represents the sources that can be mixed on links going into node i with helpers $h \in H$
a_{ij}, b_h	The cost per unit bandwidth for the edge (i, j) , The cost per unit storage for the helper h
z_{ij}, b_h	The network bandwidth usage for the edge (i, j) , The storage usage for the helper h
$f_{ij}^{(d, m_s)}$	The flow variable passing through the edge (i, j) of the connection (d, m_s)
$\sigma_i^{(d, m_s)}$	The net flow rate of a node i if the source process is m_s and the receiver is d .

Table 2: Table of Notations for the Algorithm

Symbol	Definition
$f_{ij}^{(d,m_s)}$	The flow variable passing through the edge (i, j) of the connection (d, m_s)
$\{f^{(t,m)}\}, \{f^{(h,m)}\}$	The set of unit flow paths from sources to receivers, The set of unit flow paths from sources to helpers
F	A finite field such that $ F \geq \max(T , H)$
S', s'	The set of imaginary nodes added into V connecting S , an imaginary node
$(s', s)_{m_s}$	An edge connecting the pair (s', s)
$\bigcup_{m \in D(d)} B(s', s)_m$	the information subspace be requested by the destination d
B_d	A collection of global coding vectors for each destination $d \in T \cup H$, such that this collection of coding vectors spans the information subspace be requested by the destination when stepping through each node in the network by upstream-to-downstream manner
$B(i, j)$	The collection of all global coding vectors passing through the edge (i, j)
$B_\gamma(i, j), B_\psi(i, j)$	The collection of global coding vectors passing through the edge (i, j) if $m \in \gamma$ and $f_{ij}^{(t,m)} = 1$, the collection of global coding vectors passing through the edge (i, j) if $m \in \psi$ and $f_{ij}^{(h,m)} = 1$,
S_T, S_H	The collection of receivers where $f_{ij}^{(t,m)} = 1$ The collection of helpers where $f_{ij}^{(h,m)} = 1$

List of Figures

3.1	Architecture Comparison between Existing Works.	28
4.1	Internet of Things Reference Model. Adapted from [39].	32
4.2	A Proposed Architecture of Joint Source, Network Coding Theory and Linear Optimization-based Distributed Multilevel Storage Sys- tem over OpenFlow Network for Visual Surveillance in ITS. To Ap- pear in IEEE Internet Computing Magazine [106].	33
5.1	A Collection of Interconnected Scale-free Networks. In our hybrid approach, we apply deterministic network coding for a <i>within-cluster multicasting group</i> and apply random network coding for a <i>cross- cluster multicasting group</i> . A within-cluster multicasting group in- cludes a source and all the collocated data centers; a cross-cluster multicasting group include a source and all the non-collocated data centers at the target cluster.	52
6.1	The Scale-free Networks Generated by BRITE in Our Simulation. . .	57
6.2	Implementation of Network Coding Module in UDP/IP model.	60
6.3	Software Architecture of Integrating Network Coding Module and MPEG-4 Trace File Helper over Simulated BRITE Network by Using ns-3.	62
6.4	Comparison of Network Throughput between DLNC vs RLNC across Different BRITE Networks against Time. Given a network topology, we first fix the network bandwidth (100Mbps) and packet loss rate (1%) across all links. Then we pump the data at a constant bit rate (CBR) from sources to sinks to find the network saturated point. . .	67

6.5	Comparison of Network Usage between DLNC vs RLNC at All Links of BRITE Network against Time.	68
6.6	Accumulated Network Usage at All Links of BRITE Network against Time. In this experiment, we send the video trace file <i>Verbose_silence.dat</i> [93] from sources to sinks to show the network usage.	69
6.7	Total Tags Transmitted across Network.	70
6.8	Comparison of Network Throughput of Hybrid Network Coding against All-random and All-deterministic Schemes under Scenario 1 for the Network of 808 Nodes. In the scenario 1, 5% of nodes go down randomly at the start of every second and come up at the end of the second, repeatedly.	73
6.9	Comparison of Network Throughput of Hybrid Network Coding against All-random and All-deterministic Schemes under Scenario 2 for the Network of 808 Nodes. In the scenario 2, the network is static before 10 seconds; after 10 seconds, 5% of nodes go down and leave the network at some point of time.	74
6.10	Comparison of Network Throughput of Hybrid Network Coding against All-random and All-deterministic Schemes under Scenario 3 for the Network of 808 Nodes. In the scenario 3, the network is static before 10 seconds; after 10 seconds, 5% of nodes go down and leave the network at some point of time, and 5% of nodes go down randomly at the start of every second and come up at the end of the second, repeatedly until 30 seconds.	75
6.11	Comparison of Network Throughput of Hybrid Network Coding against All-random and All-deterministic Schemes under Scenario 1 for the Network of 1208 Nodes. In the scenario 1, 5% of nodes go down randomly at the start of every second and come up at the end of the second, repeatedly.	77
6.12	Comparison of Network Throughput of Hybrid Network Coding against All-random and All-deterministic Schemes under Scenario 2 for the Network of 1208 Nodes. In the scenario 2, the network is static before 10 seconds; after 10 seconds, 5% of nodes go down and leave the network at some point of time.	78

6.13	Comparison of Network Throughput of Hybrid Network Coding against All-random and All-deterministic Schemes under Scenario 3 for the Network of 1208 Nodes. In the scenario 3, the network is static before 10 seconds; after 10 seconds, 5% of nodes go down and leave the network at some point of time, and 5% of nodes go down randomly at the start of every second and come up at the end of the second, repeatedly until 30 seconds.	79
6.14	Comparison of Network Throughput of Hybrid Network Coding against All-random and All-deterministic Schemes under Scenario 1. In Scenario 1, we assume that there are 5 nodes per cluster (the total is 20 nodes) that are down/up per second.	81
6.15	Comparison of Network Throughput of Hybrid Network Coding against All-random and All-deterministic Schemes under Scenario 2. In scenario 2, after 10 seconds, we assume that there are 5 nodes per cluster (the total is 20 nodes) that go down and leave the network.	82
6.16	Comparison of Network Throughput of Hybrid Network Coding against All-random and All-deterministic Schemes under Scenario 3. In scenario 3, the network is static before 10 seconds; after 10 seconds, 5 nodes per cluster go down and leave the network at some point of time, and 5 nodes per cluster go down randomly at the start of every second and come up at the end of the second, repeatedly until 30 seconds.	83
6.17	Comparison of Network Throughput of Hybrid Network Coding against All-random and All-deterministic Schemes when No Packet Loss. . . .	85
6.18	Comparison of Network Throughput of Hybrid Network Coding against All-random and All-deterministic Schemes under 0.05 Packet Loss Rate.	86
6.19	Average Network Throughput over Time against Packet Loss Rates. .	87
6.20	Network Throughput against Time for Data transmissions Using Hybrid Approach under Different Packet Loss Rates (no loss, 0.05, 0.10).	89
6.21	Network Throughput against Time for Data transmissions Using All-random Approach under Different Packet Loss Rates (no loss, 0.05, 0.10).	90

6.22 Network Throughput against Time for Data transmissions Using All- deterministic Approach under Different Packet Loss Rates (no loss, 0.05, 0.10).	91
--	----

Chapter 1

Introduction

Traffic surveillance [1, 2, 5] has become a hot topic in *intelligent transportation system* (ITS) with the goal of monitoring and controlling the traffic flow and events over the entire transportation system of either a city, a state, or a country, so that the whole transportation system can operate efficiently for the welfare of traffic users. During traffic surveillance and monitoring, vast amount of data is produced from a variety of sources, e.g., cameras, laser radar, GPSs, inductive-loop detectors and so on [8, 9, 10], which are distributed geographically across multiple regions. Near real-time availability of traffic data must be assured, so that the traffic predictions and forecasts can be as accurate as possible [6, 7]. Thus, it is important to design an efficient storage infrastructure that is tailored-made for the visual surveillance in ITS.

The natural choice to deal with the scale is to deploy cloud computing for ITS [4], but as the volumes of data continue to increase, we argue that it is essential to augment the centralized backend with edge resources (fog computing [17, 18]). In this thesis, we aspire to provide a solution based on the idea of fog computing for the distributed multilevel storage system in ITS which can make the usage of network bandwidth and storage become more efficient; specifically, we would like to improve the throughput of video data transmissions within a collection of clusters (local networks) where each cluster is governed by an autonomous system having the knowledge of the cluster. Most of the traffic surveillance devices are deployed at the network edges and the generated data has to be transmitted across the network from those devices to the data center for further processing. This causes inefficient usage of network resources. Moreover, the latency of the responses to the requests

of traffic users increases due to the back-and-forth transmissions if the requests are not processed locally on the spot. Fog computing extends the traditional cloud by delegating (some) tasks to the edge resources. Characteristics of fog computing, such as location awareness, low latency, geo-distribution, are suitable to support the ITS services. Furthermore, since the internet is not a dedicated network, visual surveillance services must share the available network resources (e.g., storage and network bandwidth) with other applications; hence, it is important to adjust the coding rate, data reliability scheme and network delivery path to maximize the experience of network stakeholders (e.g., network service providers, owners of data centers or edge clouds, traffic users, etc) involved in the system. Therefore, it is a crucial task to design a customized cost model for the application of fog computing in the visual surveillance of ITS when integrating the edge resources with the capabilities such as data recombining and storing so that Quality-of-Service (QoS) can be ensured for all the network stakeholders.

Recent development in network streaming combines the idea of OpenFlow [24] and multimedia multicast streaming [35] so that streaming management can be handled efficiently. Openflow is one of the realizations of software-defined network (SDN) [21, 22, 23]. In the architecture of SDN, there is a controller (residing in the control plane, which can be logically centralized but physically distributed over several servers [26, 27, 28]) that can exert control logic through the control flow on the network switches (residing in the data plane) since it has the global view of the network by collecting the feedback from network devices. The communication between control plane and data plane can be implemented by using TCP protocol to assure the control and feedback delivery while the data transmission in data plane can be implemented by using standard streaming protocols. Additionally, the SDN controller can function as a *network operating system* [25] to control the underlying network resources (e.g., network links and switches, storage devices, computing nodes, etc). Based on this, we advocate a joint source [44, 46, 47, 48], network coding [53, 54, 55, 56, 57] and optimization theoretic approach [70, 71] over distributed multilevel storage infrastructure (DMSI) based on software-defined framework in which we can design a customized model running atop the network operating system (see Chapter 4 for the details).

In the scenarios of multimedia applications, the challenges are large amounts

of data, stringent timing constraints, heterogeneity in the importance of video layers/packets as well as in the client devices; these challenges also exist in visual surveillance for ITS. Moreover, even though network coding can offer benefits such as higher throughput, saving up network bandwidth, resilience to link failures and providing a simple security mechanism by linearly combining the source packets against wiretapping attacks, these gains do not come for free. Network coding requires intermediate processing at the nodes along the path from sources to the sink¹, which may lead to additional delay, network complexity, and vulnerability to byzantine attacks [68]. Hence, the network coding algorithms have to be specifically designed to deal with these challenges under real world constraints and settings for the visual surveillance in ITS while enjoying the benefits bring by network coding.

The designed network coding solutions for real-time visual surveillance ideally should have the following three properties [52]. First, the decoding complexity must be reasonable for real-time multimedia applications. Second, the network coding algorithms must be scalable and robust against network dynamics, including the changing source rates, network churn, join-and-leave of intermediate nodes, link failures and delays, and varying bandwidth. Third, the network coding solutions must be adaptive to accommodate the various timing constraints in distributing the media contents. We intend to include all the properties in our solution.

Since there is no global view of the fast evolving network, on which the deterministic network coding is relying upon, research community advocates the use of random network coding [55, 56, 57] to design a practical distributed network coding implementation [57]. Network coding [53, 54, 55, 56] enables coding operations in the intermediate nodes between the sources and sinks, and thus increases the network throughput of data transmission compared to the traditional store-and-forward routing solution, which may be applied in the design of the visual data transmission and storage architecture for ITS. However, there are three issues associated to random network coding when applying it:

1. There is bandwidth resource loss due to the probabilistic decoding nature of random network coding where the local coding coefficients are generated randomly.

¹The words ‘sink’ and ‘receiver’ are used interchangeably.

2. There is an additional overhead (3%-6%) for tagging the generated random coefficients to the coded packets [57, 78] for constructing the decoding matrices at the sinks. For a large volume of data², as is the case for traffic surveillance applications, this additional overhead can be significant extra traffic in absolute.
3. Random network coding requires a larger field to assure higher decoding probability, which in turn increases the re-encoding complexity at the intermediate nodes [58] and decoding complexity at the sinks due to the matrix inversion [60]. The experiment results in [59] show that the overhead of random network coding cannot be neglected.

Hence, random network coding may not be appropriate for real-time visual surveillance in our proposed DMSI; other than the original issue of packet loss over the network in real world scenarios, there will be additional capacity wastage and computational complexity due to random network coding.

From the previous discussion, we can expect that both network coding paradigms perform differently in our multilevel storage architecture when deployed on the networks at different scales. For the networks at small to medium scales, the probabilistic decoding nature of random network coding may burden the network load since a lot of retransmissions are required in the delivery scheme to combat the issue of linear dependency between coding coefficients. Moreover, we cannot neglect the tag overhead and the computational complexity in the case of traffic surveillance applications. It follows that the deterministic network coding is a more economical option for the networks at small to medium scales; relying on the goodness of SDN, where the centralized controller has the global view of the network system, we can use the deterministic network coding to find the global optimum for the network system at this scale. However, the synchronization between routers in the network for the deterministic network coding may be costly for large networks. Therefore, deterministic network coding may perform better on the networks at small to medium scales, while random network coding may perform better on the networks at medium to large scales. Thus, rather than depending solely on either a random or determin-

²Cisco predicts that a smart city of 1 million will generate 180 million gigabytes of data per day by 2019 where an intelligent transportation system will take a substantial amount of the data [11].

istic solution, we propose a hybrid approach where both deterministic and random variants are used in parallel in different parts of the network for adapting to the network dynamics.

The internet is divided into a large number of different regions of administrative control commonly called *autonomous systems* (ASs). While a specific routing protocol³ is used within an AS, Border Gateway protocol (BGP) has been the de facto inter-autonomous system routing protocol [12]. It is used by BGP routers associated with different ASs to exchange the reachability information, and determine the routes for packets traversing through multiple ASs. Several researchers have studied the problems of BGP, such as routing instability [13, 14], slow convergence [15], and scalability issues [16]. These problems can lead to packet loss across autonomous systems. In [15], Labovitz et al. found that BGP convergence was delayed three minutes on average, up to fifteen minutes in extreme cases, due to path enumeration. During this time, data traffic suffered increased packet loss where the average packet loss increased from less than 1% to 30% at peak. In contrast, OSPF and IS-IS are fast-converging protocols, thus the network of an AS can achieve stability quickly. Since the failure rate of the infrastructures and devices in every AS is different, we focus on the aspect of protocols and assume that the transmission within an AS is more stable than the transmission across multiple ASs.

By assuming that we have a collection of connected clusters of nodes (each cluster models an AS), we employ the deterministic network coding for the video transmissions within a cluster and the random network coding for the video transmission across the clusters. The rationale that we use the deterministic network coding for the video transmissions within a cluster in our hybrid approach is because of the small network change (say 1% or less) in clusters (local networks) and thus the clusters are stable in the real world. We assume that the size of the network is at the scale of a city (a small to medium-sized network), so the latency between the centralized controller and the switches is negligible. To that end, we have done several simulations to evaluate the performance of the hybrid approach under different network scenarios over a collection of clusters. Our simulation results show that the proposed hybrid approach performs about 5% better than all-random ap-

³E.g., Open Shortest Path First (OSPF), Routing Information Protocol (RIP) or Intermediate System to Intermediate System (IS-IS)

proach⁴ with the finite field $GF(256)$ in terms of total network throughput when we assume that the network change happens to the nodes connecting these clusters. (see Chapter 6 for the simulation model and results).

Our contributions (see Section 3.2 for the details) can be summarized as follows:

1. We propose a cost model running atop an SDN controller to capture the network usage (e.g., network bandwidth and storage) with the aid of storage helpers for the network stakeholders. In our design, there is a clear separation between the helpers' flows and receivers' flows, which allows the consideration of adding the functionality of local repairable codes for the layer of distributed storage helpers to provide reliability for the cached video data.
2. Based on the knowledge about cluster provided by the SDN controller, we emphasize on a hybrid approach in which the data transmissions within a cluster are empowered by deterministic network coding while the data transmissions across clusters are empowered by random network coding. To evaluate our proposed hybrid approach, we create a simulation platform by using ns-3 and we use this platform to validate our claim. As mentioned in the previous paragraph, the proposed hybrid approach performs about 5% better than the all-random approach when the network change happens to the nodes connecting the clusters. This is important since the setting fits into realistic scenario where the data transmission within a local network is stable while the data transmission between clusters is less stable.

With this in mind, we explore several technologies and concepts relating to fog and cloud computing, software-defined network and network coding for the underlying physical and network layers of a DMSI in ITS in Chapter 2. Next, we highlight the related works and our contributions in Chapter 3. Then, we formulate goals, core functionalities and challenges of DMSI for visual surveillance in ITS in Chapter 4 and the system model in Chapter 5. Specifically, we describe the system model and formulate the video data transmission and storing as a linear optimization problem (LO) with minimum resource usage cost subject to the constraints of several system parameters, i.e., network bandwidth and storage, in Section 5.1. After that, we give

⁴For the all-random approach, we apply random network coding, either with $GF(16)$ or with $GF(256)$, for both within-cluster multicasting and cross-cluster multicasting.

the algorithm for constructing a network code to set up the network for every feasible solution in the feasible region of the aforementioned LO problem in Section 5.2 and provide the claims and proofs in Section 5.3. In Section 5.4, we discuss our hybrid approach and model for ITS. In the following chapter, we provide our simulation results. Finally, we conclude and elaborate our future works in Chapter 7. For the ease of reading, we also summarize the notations used in this report in Table 1 and Table 2.

Chapter 2

Background Technologies and Concepts

In this chapter, we briefly introduce the required technologies and concepts which are useful for the conceived distributed multilevel storage infrastructure (DMSI) for intelligent transportation system (ITS). Since cloud computing provides an efficient way of manipulating geographically distributed traffic devices and resources through virtualization and distributed computing techniques [4], we can expect that cloud computing is an essential and indispensable component in our proposed architecture. Moreover, large amount of data generated from multiple heterogeneous sources has changed ITS from a conventional technology-driven system into a more powerful multifunctional data-driven system [5]. Therefore, we will give an overview of the cloud computing and data-driven intelligent transportation system (DDITS) in Section 2.1. To deal with the large amount of data generated by the data-driven system located at the network edge, it is vital to extend centralized backend with edge resources [17, 18]. Thus, we provide an introduction to fog computing and software-defined networks in the next section. In Section 2.3, we will describe the intelligent visual surveillance system, which is a component of DDITS. Note that the usage of these technologies and concepts span across multiple network layers, from the lowest (physical layer), to the highest (application layer). We also provide an introduction to network coding theory, which is the theoretical ingredient for DMSI, in Section 2.4. Finally, we will describe the software tools used in our simulation in Section 2.5.

2.1 Distributed System in ITS and Cloud Computing

Cloud computing, which relies on several technologies such as utility, grid, distributed and parallel computing, service-oriented architecture, web services, data centers, virtualization, etc., provides elastic computing resources over the network to users based typically on pay-as-you-go model. Deploying ITSs over a cloud platform enables manipulating, provisioning and storing large amounts of data from diverse and disparate sources. However, this cloud-computing paradigm is accompanied with several challenges when deploying in real world scenarios, such as user admission control, resource scalability and optimal utilization and workload balancing across virtual machines. The work in [4] surveyed several distributed systems for ITS in literature, for example, Internet-based Advanced Traveler Information Service and On-demand Bus Service.

By considering the large volume and variety of traffic data that is generated from diverse sources and has to be frequently updated and archived, a data management framework catering to ITS is necessary. This has changed ITS from a conventional technology-driven system into a more powerful multifunctional data-driven intelligent transportation system (DDITS) [5], which is a vision-driven, multisource and learning-driven framework for ITS that can be realized on top of cloud platform. The difference between DDITS and conventional technology-driven ITS is that the conventional ITS put more focus on the historical and human experiences, but places less emphasis on the utilization of real-time ITS data or information. In fact, DDITS is a system framework that would:

- Allow users to interactively utilize data resources that pertain to transportation system.
- Access and employ data through more convenient and reliable services to improve the performance of transportation system.
- Realize and extend the functions of core services of ITS.

A DDITS system has three main components, which are vision-driven devices (such as intelligent surveillance systems), multisource-driven interface (where the

data can be collected from diverse sources), and learning-driven applications (where the combined techniques of data fusion and online machine learning are used in traffic control and management). For DDITS to be widely accepted, the system must be privacy aware and people-centric [5], emphasize more on the utilization of real-time ITS data[6, 7], and increase the number of data sources during data collection [8, 9, 10]. In fact, visual surveillance and the subsequent intelligent processing and data-intensive learning can be considered as a part of DDITS.

2.2 Fog Computing and Software-defined Concepts

Fog or edge computing [17, 18] extends the cloud computing paradigm by pushing compute, storage, and networking services away from centralized nodes such as data centers, to the edge nodes of a network. In ITS, local surveillance systems connecting to wired or wireless network can be identified as edge nodes. Characteristics of fog such as location-awareness, low latency and geo-distribution are suitable to support massively distributed data generation, as well as data and service requests, particularly since the requests mostly concern the local data itself. Moreover, edge application services significantly reduce the amount of data that has to be transmitted from one node to another by local pre-processing at the edge nodes, and thus decrease transmission costs, shrink latency, save network bandwidth, and improve quality of service. This is because the consequent traffic and the distance the data must travel is reduced correspondingly. Hence, it is natural to anticipate the application of fog computing in the area of ITS.

In the meantime, the demands on the network is increasing. In addition, the complexity of current networks due to the isolated development of different protocols intended to solve a diverse set of problems over last decades has complicated the situation. Moreover, the current gigantic network scale has resulted in incredibly dynamic and unpredictable traffic patterns and has caused the traditional network solutions unable to scale well. Traditionally, the control plane (an abstraction layer, making decisions where the traffic is sent) and the data plane (an abstraction layer forwarding traffic to the selected destination decided by control plane) are combined together in each network node [21, 22, 23]. This has proven restrictive for flow management [21, 29, 30, 31], since the only way to make adjustments is reconfiguring

the network devices once the initial forwarding policies have been defined. Thus, network operators are unable to respond quickly to the fast-paced changes of network traffic and the impact of big data [29, 30]. These trends have pushed the search for a viable solution.

Software-defined networking (SDN) has come to the fore for addressing the dynamic nature of network functions and intelligent applications while lowering operating costs through abstraction of lower level functionality and simplified software, hardware and management. This is done by separating the control plane from the data plane, which allows network administrators to manage and provision network services through the control plane that is logically centralized. OpenFlow protocol is one of the realizations of SDN. SDN is leveraged to design new systems. For example, NOX is a *network operating system* [25] built on top of OpenFlow switches that provides a uniform and centralized programmatic interface to observe and control the entire network, which is analogous to read and write access to various resources provided by computer operating systems. In the design of NOX, there is a southbound connection connecting the control plane NOX controller with the OpenFlow switches (OpenFlow protocol) in the data plane, and a northbound connection connecting the NOX controller with network services and applications in the application layer.

The challenges and opportunities of implementing SDN are discussed in detail in [21] and [23]. The former focuses on the issues of network programmability for performance, scalability, security, and interoperability. The later delves into the design architecture and issues of the SDN controller, the connection and communication protocols between the controller and switches in the data plane (southbound APIs, e.g., OpenFlow) and between the controller and services (northbound APIs).

2.3 Intelligent Visual Surveillance System

The main goal of an intelligent visual surveillance system is to provide an automated interpretation of real-time visual information generated from deployed cameras and sensors that capture either persistent or transient objects, and forecast the dynamics of observed objects. The main stages in the image processing framework of an intelligent automated surveillance system include moving object detection and recogni-

tion, classification, tracking, behavioral analysis, and data fusion and retrieval. For a comprehensive treatment of these aspects we refer the readers to [1, 2, 3].

Reliable and robust vehicle detection is the first step of traffic video processing, which aims at segmenting regions corresponding to observed and moving vehicles from the background, and important for the subsequent processes such as vehicle tracking, recognition and behavioral analysis. There are two main categories in vehicle detection, which are appearance-based methods and motion-based methods. Appearance-based methods use features such as color, texture, and shape to detect vehicles while motion-based methods distinguish vehicles from the stationary background by using moving characteristics of the vehicles. There are several main challenges arising from the detection research, such as vehicle shadow, vehicle occlusion, and nighttime detection. After vehicle detection, the surveillance systems track moving objects from one frame to another in an image sequence, based on the assumption that vehicles in two adjacent frames are the same if the spatial distance is small. Tracking over time involves matching objects in consecutive frames using features such as points, lines, or blobs. Useful mathematical tools for tracking include Kalman filter, condensation algorithm, dynamic Bayesian network, geodesic method, etc.

2.3.1 Evolution of Surveillance Systems

In the first-generation analog CCTV systems, video streams are generated by cameras in analog format and then gathered, switched and dispersed by the analog matrix switch to analog video cassette recorders for recording or to display screens for monitoring. Such traditional CCTV systems are independent of and disconnected from the internet. Over years, surveillance systems have evolved from traditional analog CCTV systems to sophisticated (fourth-generation) wide-area surveillance systems which are distributed and network-centric [2], and reliant on the internet. Network-centric surveillance systems use cameras that are Internet Protocol (IP) enabled to transmit digital data over an Ethernet or wireless network. [2] had summarized many distributed surveillance systems from the literature in ITS. The advantages of using IP network-centric surveillance systems over analog CCTV systems include reduced cost, higher resolution, scalability, secure remote access via VPN or HTTPS and unlimited data storage through cloud storage.

However, it is difficult to integrate data from distributed sources. Moreover, most of the existing distributed surveillance systems are composed of many one-camera local systems [2], which can only be used to analyze local events, but lacking of efficient technologies for whole scale transportation network monitoring (e.g. lacking efficient algorithm for predicting global road traffic). Therefore, designing a DMSI relying on the interplay of fog and cloud, which is used to support the global transportation monitoring and controlling solutions, looks promising. In addition, in terms of communication and integration of different layers, it is necessary to study new communication protocols and metadata standard, which should be more efficient in resource usage and allocation that is tailor-made for ITS.

2.4 Network Coding Basics

Network coding [53, 54, 55, 56, 57] is a novel technique which enables an information source simultaneously transmits the same information to multiple receivers in the network by allowing the generated information to be encoded at the source, re-encoding at the intermediate nodes along the paths from the source to the sinks, and decoding at the sinks over arbitrary network. The traditional store-and-forward mechanism follows the Kirchoff's law, where the sum of information bits flowing into an intermediate node is equal to the sum of information bits flowing out of the node. For network coding, Kirchoff's law doesn't apply, since network coding considers each network information flow with different destinations as algebraically combinable entity. Thus, the sum of information bits flowing out of an intermediate node can be a function of the flows flowing into that node.

This new technique greatly increases network throughput [53], significantly reduces latency and energy [63, 64], makes the network more robust, and also provides obfuscation against eavesdropping [67], hacking or other forms of attack [68].

In a linear network code [54, 55], all the information symbols, including the symbols that comprise the information source as well as the symbols transmitted on the channels, are regarded as elements of a finite field F^1 called the base field. A finite field is a set of symbols together with the associated operators (e.g., addition and multiplication), which has the same algebraic structure as the system of real

¹For example, F is taken to be the binary field $GF(2)$ when the information unit is the bit.

numbers in which we can perform daily arithmetic such as addition, subtraction, multiplication, and division. Unlike the real number field that has an infinite number of elements, a finite field has only a finite number of elements. Since we don't use the detailed structural properties of finite fields in the subsequent discussion, one can regard the operations in finite fields work as the same as the operations in the system of real numbers. For the theory of finite fields, we recommend the book written by Lidl and Niederreiter [40].

2.4.1 Network Coding Model for Single Source Multicasting

To begin with, we consider a single source multicasting model in which there is a communication network represented by a directed acyclic graph $G = (V, E)$ with an information source $s \in V$ and a set of receivers $t \in T \subseteq V$ where $s \neq t$ and each edge $e \in E$ has unit capacity, i.e., $c(e) = 1$, which means that each edge can carry one symbol per unit of time. Each symbol is an element of some finite field F and each receiver $t \in T$ requests all information of s .

A *cut* between s and $t \in T$ is a set of edges whose removal disconnects² s from t . The *value* of a cut is the sum of the capacities of the edges in the cut. A *min-cut* is a cut with the minimum value. We denote $MinCut(s, t)$ to be the value of a min-cut between the source s and a sink $t \in T$. We will use the term min-cut to refer to both the set of edges and to their total value. Note that $MinCut(s, t)$ is unique and there are possibly several min-cuts.

By max-flow min-cut theorem [20], the information can be sent from s to $t \in T$ at a maximum rate of $MinCut(s, t)$. In other words, the number of edge disjoint paths between s and $t \in T$ is equal to $MinCut(s, t)$. Therefore, to multicast the same information to all the receivers $t \in T$, we must have

$$h = \min_{t \in T} MinCut(s, t). \quad (2.1)$$

Let $x_1, \dots, x_h$ be the information symbols that we want to multicast from s to T in each unit of time.

²In a directed graph, two nodes u and v are *connected* if there exists a directed path from u to v or from v to u .

2.4.2 Local and Global Encoding Vectors

In linear network coding, the coded symbol emanating from a node $v \in V$ through an edge $e \in E$, denoted as $y(e)$, can be computed as a linear combination of the symbols $y(e')$ on edges $e' \in E$ entering the node v . That is,

$$y(e) = \sum_{e' \text{ entering node } v} \alpha_{e'}(e)y(e') = \boldsymbol{\alpha}(e)\mathbf{y}(e')^T, \quad (2.2)$$

where $\boldsymbol{\alpha}(e) = [\alpha_{e'}(e)]_{e' \text{ entering node } v}$ and $\mathbf{y}(e') = [y(e')]_{e' \text{ entering node } v}$ are row vectors. The coefficients $\alpha_{e'}(e)$ of the linear combination forming the vector $\boldsymbol{\alpha}(e)$ is known as *local encoding vector*. The local encoding vector $\boldsymbol{\alpha}(e)$ on edge e leaving node v specifies the network operations performed on the symbols entering node v . The length of $\boldsymbol{\alpha}(e)$ is determined by the number of edges e' entering v .

Now, we would like to express the coded symbol $y(e)$ on any edge $e \in E$ as some function of the source symbols $x_1, \dots, x_h$. We assume that there are h artificial unit edges $e'_1, \dots, e'_h$ entering s , each transmit one source symbol x_i , $1 \leq i \leq h$, respectively. By induction, it is clear that $y(e)$ can be expressed as the linear combination of $x_1, \dots, x_h$, that is,

$$y(e) = \sum_{i=1}^h g_i(e)x_i = \mathbf{g}(e)\mathbf{x}^T, \quad (2.3)$$

where the row vector $\mathbf{g}(e)$ is called the *global encoding vector* on edge e . By inspection, we can see that the global encoding vector $\mathbf{g}(e)$ on edge e emanating from v can be computed recursively as

$$\mathbf{g}(e) = \sum_{e' \text{ entering node } v} \alpha_{e'}(e)\mathbf{g}(e'), \quad (2.4)$$

by using the local encoding vector $\boldsymbol{\alpha}(e)$.

Suppose now that a receiver $t \in T$ receives h coded symbols $y(e_1), \dots, y(e_h)$ from edges $e_1, \dots, e_h$. The received symbols can be expressed in terms of source symbols as the following matrix equation:

$$\begin{bmatrix} y(e_1) \\ \vdots \\ y(e_h) \end{bmatrix} = \begin{bmatrix} g_1(e_1) & \cdots & g_h(e_1) \\ \vdots & \ddots & \vdots \\ g_1(e_h) & \cdots & g_h(e_h) \end{bmatrix} \times \begin{bmatrix} x_1 \\ \vdots \\ x_h \end{bmatrix} = G_t \begin{bmatrix} x_1 \\ \vdots \\ x_h \end{bmatrix}, \quad (2.5)$$

where the i -th row of the matrix G_t is the global encoding vector associated with the edge e_i entering t . If G_t is full rank, then the receiver t can retrieve the source

symbols by multiplying the inverse matrix G_t^{-1} to both sides of Equation 2.5. Therefore, choosing global encoding vectors so that all G_t , $t \in T$, are full rank will enable all receivers to recover source symbols from the coded symbols they receive. Also, the global encoding vector of an output edge of a node must lie in the linear span of the global encoding vectors of the node's input edges.

2.4.3 Main Theorem of Network Multicast

The network code design problem is to select values for the entries of local encoding vectors so that all matrices G_t are full rank. The single source multicast theorem of Ahlswede et. al. [53] can be expressed in algebraic language as follows:

Theorem 1 *In linear network coding, there exist values in some large enough finite field F_q for the entries of the local coding vectors, such that all matrices G_t , $t \in T$, defining the information that the receivers observe, are full rank.*

Proof The proof is omitted. For the readers interested in the proof, we highlight the book by C. Fragouli [49, Theorem 2.2].

2.5 Software Tools

In this section, we introduce the software packages and libraries being used in our simulation.

2.5.1 ns-3

The ns-3 simulator [94] is a discrete-event network simulator targeted primarily for research and educational use. In brief, ns-3 provides the models of how packet data networks work and perform, and provides a simulation engine for users to conduct simulation experiments. ns-3 is mainly supported for Linux, OS X, and FreeBSD and is under the GNU GPLv2 license. Note that ns-3 is not backward compatible with its predecessor, ns-2.

The reason we choose to use ns-3 is that we want to investigate the video transmissions over large SDN network by using network coding methods, which is difficult to perform with real systems at scale. This difficulty arises both in terms of having access to such an infrastructure, and also from the impracticality of testing each idea

and hypothesis over a real infrastructure in a controlled and reproducible manner. Also, we want to study the behavior of the system in a highly controlled and reproducible environment, such as ns-3. Moreover, ns-3 provides an additional module to set up SDN/Openflow network and an interface to integrate with BRITE [95, 96].

2.5.2 BRITE

BRITE (*Boston university Representative Internet Topology gEnerator*) [95, 96] is a universal topology generator which adheres to the following three design principles: representativeness, inclusiveness, and interoperability. Representativeness allows the users to construct synthetic topologies that accurately reflect many aspects of the actual Internet topology (e.g. hierarchical structure, degree distribution, etc.). Inclusiveness combines the powers of different generation models in one single generation tool. Interoperability provides interfaces to widely-used simulation applications such as ns-3 and OmNet++ as well as visualization applications.

By coupling BRITE with ns-3, we could create large Internet topologies quickly and efficiently, while taking advantage of ns-3’s simulation capabilities. There are several works [97] that have leveraged on such a close integration to simulate large networks. We will further describe the assumptions and parameters we have chosen for the generation model in subsequent Section 6.1.1.

2.5.3 Kodo

Kodo [88] is an open source C++ library for implementing erasure correcting codes, in particular, network codes. The library is intended to be used as a building block for reliable communication protocols and systems, and for research on the implementation of network coding methods. To provide a research-friendly library Kodo provides a number of algorithms and building blocks, which includes random linear network coding (RLNC), Reed-Solomon codes (RS), etc. In general, the implementation of RLNC follows the idea from Chou et. al. [57]. Furthermore, it is possible to integrate Kodo and ns-3, and test the performance of the network codes over wired and wireless networks [89].

2.5.4 MPEG-4 and Video Trace File

MPEG-4 is a method of defining compression of audio and visual (AV) digital data. The visual compression is defined in Part-2 of MPEG-4 [90] while the audio compression is defined in Part-3 [91]. The visual compression format in Part-2, which supports B-frames and other new features, is an extension of H.263. A MPEG-4 video can be 24 or 48 frames per second (fps). The group of picture (GoP) structure defining in Part-2 is IBBPBBPBBPBBI. However, the source coding method encodes the GoP in this order: IPBBPBBPBBIBB. A video trace file [92, 93] of a video is a table where each entry of the table records the frame type (either I, P, or B), the frame size (in bytes) and the generated time (in milliseconds) of video frames in the video. Also, the order of the entries in the table is corresponding to the encoding order of the video frames in the video. The video trace files provided by the website [93] is of 24 fps. By using video trace file, we can simulate the video transmissions over a BRITE-generated network without the real video content.

Chapter 3

Related Works and Our Contributions

3.1 Related Works

In order to shed some light on our proposal, we give a literature review on the recent development of several related fields. We'll discuss current work of others in the areas of network coding mechanisms, followed by network coding for streaming, networking, SDN for streaming, and simulation of network coding.

3.1.1 Network Coding

Network coding enables information symbols from different flows to be combined at intermediate nodes of the network to reduce the network bandwidth, and so the application of network coding in the current deployment of network models has the potentials to save up a tremendous amount of network bandwidth and increases the network throughput over time. Ahlswede et al. [53] pioneered the field of network coding, studied the problem of single source multicast scenario and characterized the coding rate region for the network information flow. They showed that the source can multicast information to all receivers at a rate approaching the source rate as long as the size of the block code is sufficiently large, which can be regarded as the generalization of Max-flow Min-cut theorem for information flow. Subsequently, Li et al. [54] used the tools from graph theory and linear algebra to describe an optimal model for the directed acyclic graphs and showed that linear coding, where

the information symbols are combined linearly (finitely many symbols) at the nodes of the graph, can be used to achieve the optimum (source rate). They also started the general framework to build the deterministic network coding and gave an exponential time algorithm for the construction. Koetter et al. [55] opened up another formulation for network coding, which is purely algebraic and based on algebraic geometry and matrix theory. This work also provided the random network coding framework, where the coefficients of the linear combination can be chosen randomly. Jaggi et al. [56] provided the first polynomial time construction for deterministic network coding, while Chou et al. [57] provided the first practical construction of random network coding by using the idea of tagging the random coefficients to coded packets, generation numbers from source and buffering of coded packets at nodes. Nevertheless, the simulated system in Chou's work is relatively small, which only consists of 89 nodes and 972 bidirectional edges; this is obviously not on a par with the scale of real world networks. Moreover, Chou also discussed the weakness of their work in [61], which highlighted the difference between the assumptions in theoretical network coding and the real networks. All of the works above can be classified as intrasession network coding, where the information symbols within the same established session between the single source to each receiver are linearly combined at nodes. This is in contrast to intersession network coding, where the restriction is lifted and information symbols from different established sessions between multiple sources to different receivers can be linearly combined.

Even though intrasession network coding is able to save up network bandwidth and increase the overall network throughput over time, it is suboptimal [62, 63]; intersession network coding (multiple unicasts or multicasts) is necessary to achieve better network throughput in general. However, performing intersession network coding is hard, and heuristic methods are used in general [63, 64]. This problem is usually formulated as a linear optimization model with various kinds of objective functions (e.g., minimize the cost of network bandwidth usage, minimize the number of hops between source to receiver, minimize the energy usage, maximize the network throughput, etc.) and graph-theoretic constraints. In [63], Lun et al. studied the problem of multiple multicast sessions, where there are multiple sources sending information symbols to a set of receivers in which each receiver requests a different subset of the source processes and applied the methods of [56] to construct a network

code for every feasible solution in the region of graph-theoretic constraints. They also discussed the possible objective functions and examined the corresponding optimal solutions. They extended their work and formulated a minimum cost model by using convex optimization to analyze the wireless and wireline packet networks in [64] so that network participants can quantize their network usage and thus decide a better network strategy. Traskov et al. [65] analyzed the multiple unicasts scenarios and described the corresponding capacity region as TRLKM region. Those works discussed so far are operating on given flow rates, Eryilmaz and Lun [66] proposed a dynamic-routing-scheduling-coding strategy that operates solely on queue-state information. Following the same line of queue-state approach, [69] proposed a simple network coding scheme for multicast applications, which optimizes multicast throughput using both intrasession and intersession network coding strategies.

For a comprehensive survey, the readers can refer to [50] for further information about network coding theory, which gives the antecedents and current known fields of theoretical network coding. Also, the applications of network coding are discussed in detail in [51] and [52]. The former focuses on the areas of throughput/capacity enhancement, robustness enhancement, network tomography, and security. The later delves into the multimedia applications, particularly on the issues of delay, in-network error control, and media-specific control.

3.1.2 Network Coding for Multimedia, Video-on-Demand and Live Streaming

As we mentioned in the last subsection, the concepts of network coding originated from information theory community; however, the gaps between the information-theoretic models and the models of multimedia applications are fairly huge and different in several aspects. First, large delays and packet losses due to network congestion are common in practical networks, and these issues must be considered with respect to the timing constraints of multimedia applications. However, timing constraints are not considered in the network coding model. Second, the computations of optimal and nearly optimal network codes usually consider that the network topology or the buffer content of network nodes is known in advance; however, in many multimedia applications, the topology is rarely known and the buffer content

changes over time. Third, the network coding model does not consider delay jitter and bandwidth fluctuations, which are challenging issues for multimedia streaming. Fourth, network coding concerns about how to increase the available bit-rate (bandwidth), while in multimedia applications the performance is evaluated in terms of rate and distortion.

Chou et al. [57] were the first to consider network coding under practical settings. Their scheme is decentralized, which obviates the need to know the global network topology and the need to synchronize the underlying network nodes, and thus it is robust against delays and link failures. They proposed to tag the global encoding vector (that is generated on each edge in the network for the incoming packets) and the generation number (for each generation of information packets from source node) on the linearly combined outgoing packet. At sink nodes, the probability that the set of global encoding vectors forming a full rank matrix for decoding is high as long as the field size is sufficiently large. Each receiver is also attached with a buffering model (with a timing constraint for each generation of information packets), which is used to cache the coded packets of the same generation. If the newly received coded packet does not increase the rank of the matrix, it will be discarded (non-innovative packet); otherwise, it will be stored in the buffer (innovative packet). When the matrix is full rank after receiving enough innovative packets, the sink will decode the stored packets to retrieve the original information, as long as the stored packets of the same generation are within the timing constraints. This proposal, however, incurs an additional rate overhead (about 3% to 6%) due to the tagged global encoding vector, whose size depends on the number of information packets per generation and the size of finite field.

In scalable video coding (SVC) [44], multiple quality layers are created in order to adapt to the different demands from clients, which exhibit graceful quality degradation under the event of packet losses. In a P -layer system (or a multi-resolution code, MRC) [46, 47], the first layer provides a baseline level in terms of video quality and/or spatial and temporal resolutions while additional enhancement layers, up to layer P , can be used to improve on quality, resolution, or a combination thereof. SVC should be combined with the techniques from erasure protection in order to provide a more reliable content distribution of the most important layers. This can be done in several ways, e.g., allocating a forward error correction code [41] with

lower code rate, more retransmission opportunities or applying priority encoding transmission (PET) [45] to the first layers. In PET, one can specify priorities for different media segments (e.g., quality layers), and packets are created as a combination of original and coded packets, such that the baseline layer can be recovered from few packets while higher enhancement layers can be decoded when more packets are received. Note that Chou et al. [57] also applied PET for the reliability protection. Multiple description coding (MDC) [48] is an alternative to MRCs that generates multiple video streams, where the “descriptions” are independently decodable and hence do not require any unequal error protection; when more than one description is available, they can be synergistically combined to enhance the quality.

In the network coding design for multimedia applications, media-specific error control handles the distortion issues in addition to coding rate. This is achieved by extending the techniques from media compression (such as MRC and MDC) and the principle of unequal error protection to network coding [72, 73, 74, 75, 76, 77, 78]. The works in [72, 73] extended the idea of unequal error protection to network coding for multimedia in a way that the video sources encode data with scalable coding principles so that the first group of packets of a compressed video contains the most important video information while additional groups, up to P many groups, are used to improve the overall quality as mentioned in the previous paragraph. Then a random network coding scheme is applied on these groups such that the first group always has higher decoding probability. This is done by creating network-coded packets that are linear combinations of packets belonging to the first i out of P groups. Inter-generation network coding had been proposed in [74] where the scheme allows the encoding among generations for the purpose of enhancing network coding decodability. The authors of [75] had suggested a method (achieving by the concatenation of a short-length delay-mitigating code and a network code) that can be applied to MRCs, which allows the first and most important layers are decoded earlier and forwarded to the application layer without any delay. However, the ability to receive and decode the layers in order comes at the expense of a larger overall overhead. Kim et al. [77] proposed a distributed message passing algorithm to generate network codes for single source multicast of MRCs, which guarantees decodability of the base layer at all receivers, by using pushback message from sinks to source. In [78], the authors showed that there is significant advantage in using

practical network coding and MDCs for video multicast in lossy packet networks. They also observed that compared to traditional routing methods, network coding provides significant improvement in the PSNR of video at the receivers.

For video-on-demand (VoD) and live streaming (LS), a lot of attempts have been devoted to address the issues arising from both realms [79]. In the realm of VoD, storage constraint becomes another optimization dimension when we consider the concept of storage helpers¹, while for LS, storage is not a constraint because every user has the same playback time and thus synchronization between different users is a more important consideration which is relying on the available network bandwidth.

Zhang et al. [79] proposed a linear optimization model to minimize the server load with the aid of a collection of helpers that have limited storage and upload bandwidth, which allows an optimal selection of helpers for each user (those helpers being selected are one hop away from the user), for VoD. In this type of helper-assisted systems, the network devices, i.e., video server, helpers, and users, are divided into 3 layers, where the video server is at the top layer, the helpers with storage capacities are in the middle layer, and the system users are in the bottom layer. The video server connects to helpers or users in one hop and the helpers connect to users in one hop. This work did not make use of the idea of network coding in the transmissions of multimedia data over the network; instead, they used the traditional routing (store-and-forward) for the data transmission over the network.

Ostovari and his collaborators [80, 81] extended the work of Zhang et al. and applied the idea of random network coding for the video transmission over the network (their linear optimization model is similar to Zhang's). In their work, the random network coding is performed on the data packets from either same layer (intralayer network coding) or different layers (interlayer network coding) of a video at the information source for once, and the coded packets are then be routed to helpers to serve the users without further coding; no more coded packets are going to be re-encoded at the intermediate nodes (network coding at source).

¹In our context, a storage helper has limited storage (e.g., edge device) while a receiver has unlimited storage (e.g., data center); both act the same as a destination node to receive video data in network coding, however, a storage helper must respect to the storage constraint in a cost model while a receiver is not.

3.1.3 Network Coding for Networking

After more than a decade of development and achieving many fruitful results in network coding, the research community starts to incorporate network coding in network protocols, e.g., TCP, software-defined network (SDN), etc., to realize real world deployment.

The researchers from MIT [82] made the first attempt to combine network coding in the TCP layer with a minimal change and measured the performance of TCP without network coding. Subsequently, Kim et. al. [83] analyzed the performance of TCP and TCP with network coding (TCP/NC) in lossy networks by extending the framework built by [84]. Their analytical results showed that network coding masks random erasures from TCP, thus preventing TCP's performance degradation in lossy networks which results that TCP/NC has significant throughput gains over TCP.

Also, there are several works [35, 36] involving the concepts of SDN [21, 22, 23] and multimedia in which the video data are multicasted from source to receivers over a dedicated network where the control logic of data flow is separated from the data plane. The authors in [35] designed an SDN-based, scalable multimedia multicast streaming framework, which optimizes multimedia flow management and provides scalability and flexibility. Their implemented prototype showed the possibilities of in-network identification, processing, and management of media streams and access control. In [36], Egilmez et. al. presented OpenQoS, which is an Openflow based prototype powered by Floodlight [38] for the dynamic QoS routing for multimedia application over real test network. They showed that OpenQoS outperforms current state-of-the-art standard when using with TCP and guarantees seamless video delivery with little or no artifacts with TCP or UDP. However, those works didn't apply the network coding techniques in the design of the framework or prototype.

3.1.4 Simulation of Network Coding

The work in [86] extended the work of Chou et al. in [57] and implemented over SDN by using mininet [37]. They argued that the probabilistic nature of random network coding would hamper the decoding performance, so they chose to use a deterministic version of network coding [56] in their realization. They also used the

idea of tagging, generation numbers and buffering at SDN’s switches from [57] to enhance their system. However, their work lacked a comprehensive evaluation on their system prototype and did not have any theoretical analysis. Szabo et al. [87] did a similar work to [86] but they changed the underlying network coding strategy to random version and made comprehensive experiments and evaluations. However, their evaluation focused on very general data and network setting, and not specified to multimedia data.

Recently, Marcano et al. [89] integrated Kodo library [88] into ns-3 [94] and measured the performance of the library for the transmission of single data generation over wired and wireless networks. Kodo library is a network-coding-enabled software package developed by Steinwurf ApS Group, while ns-3 is a popular network simulator (see Section 2.5). Their simulation results showed that the larger the field size, the less number of transmissions is required. This is because using smaller field size will create the issue of dependency when generating the coding coefficients if the number of nodes is far greater than the field size [55, 56]. However, their work didn’t consider the impact of network coding on video transmission.

3.2 Our Contributions

To multicast traffic video data from camera sources to data centers over the internet, the visual surveillance services must share the available network resources (e.g., storage and network bandwidth) together; hence, it is important to adjust the coding rate, data reliability scheme and network delivery path to maximize the network experience of every party involved in the system. As a consequence, it is an important task to design a customized cost model for the application of fog computing in the visual surveillance of ITS when integrating the edge resources such as computing and storage devices. To design such a cost model tailor-made for integrating the storage helpers in visual surveillance over SDN networks for IoT, we propose to apply network coding to combine the data flows generated by different camera sources (see Section 4.1 and Section 4.2 for the design architecture and Section 5.1 for the cost model). However, as shall be mentioned in Section 5.1, the problem of determining whether or not a set of multisource-multisink connections is feasible with network coding still remains unsolved [63, 64]. Nonetheless, we can still find

non-trivial cost improvements by using network coding in our construction. Inspired by Jaggi’s and Lun’s works [56, 63, 64], we extend their constructions and include storage helpers (idea from VoD and edge computing) in the network system design for visual surveillance in ITS; the system, hence, is able to cache video surveillance data at those helpers during the peak hour. Those storage helpers are close to the source processes (e.g., visual cameras) and form a layer of “edge storages” to store the video data that is not requested by the data centers, and thus can save the network bandwidth between this layer and the data centers. Accordingly, we include a set of storage constraints into the set of graph-theoretic constraints in the linear optimization model to reflect the storage capability of those helpers (see Equation 5.6).

For every feasible solution in the feasible region of the cost model (the LO problem in Section 5.1), we also design an algorithm (see Section 5.2) to construct deterministic network code for efficient streaming and buffering of the visual surveillance data. There are two types of information flows in the model, one is the set of flows from sources to receivers, and another is the set of flows from sources to storage helpers; we assume that both sets of flows are never overlapped, where the first set is for real-time video transmission (see Algorithm 2), and the second set is for video caching on storage helpers during peak hour (see Algorithm 3). For the second set of flows, we can install the recently discovered distributed storage codes, such as regenerating codes [103] or locally repairable codes [104], on the storage helpers to capture the distributed property of those edge components. These codes provide better storage-bandwidth tradeoff than the traditional erasure codes when repairing lost data fragments.

Here, we compare the difference between our proposal and various other existing works in Figure 3.1 to highlight our features.

Moreover, we emphasize on a hybrid approach for networks at different scales, rather than depending solely on either a random version or a deterministic version, due to the following reasons:

1. Both network coding paradigms may perform differently in our multilevel storage architecture when deploying on the networks at different scales; deterministic network coding may perform better on the networks at small to medium scales while random network coding may perform better on the networks at

Solution	with SDN	with NC	Applications/targets	Notes
Our solution	Yes, a logically centralized controller with global knowledge	Yes, aspire to develop a hybrid approach	Real-time visual traffic surveillance in Intelligent transportation system	Emphasizes on a helper-assisted multilayer storage system where the storage helpers are close to traffic surveillance devices and an optimization cost model atop SDN controller to optimize the system performance.
[19]	Yes	No	Video Streaming	Emphasizes on using the openflow-table to control the video flows by using a centralized controller. The system different port numbers to distinguish the sources.
[48]	No	Yes	General	Emphasizes on a heuristic approach with linear optimization model, queue-based approach (NC)
[60]	No	Yes, random NC	VoD/Live streaming system	Emphasizes on a helper-assisted storage system where the storage helpers are close to system users and an optimization model to optimize the system performance.
[65]	Yes, a centralized controller (openflow) with global knowledge of network	Yes, deterministic NC based on the global knowledge	General	Only a high-level idea without any proper evaluation of the system performance on the important aspects such as overhead/gain of their proposal vs traditional methods; it is more like a vaporware.
[66]	Yes, a centralized controller combined with network function virtualization (NFV)	Yes, random NC	General	Compares the performance of their prototype (NC) with block codes in a "minimalistic" network. The experiment setup is divided into 3 cases: block codes (fountain codes) end-to-end, block codes (fountain codes) hop-by-hop, and RLNC.

Figure 3.1: Architecture Comparison between Existing Works.

medium to large scales.

2. For the networks at small to medium scales, the probabilistic decoding nature of random network coding may burden the network load since a lot of re-transmissions are required in the delivery scheme to combat the issue of linear dependency between coding coefficients.
3. There is a data overhead (about 3% to 6%) for attaching the random coefficients to the coded packets at the intermediate nodes for the decoding purpose at the receivers. For humongous volume of data, as is the case for traffic surveillance applications, this additional overhead can be significant extra traffic in absolute.
4. Relying on the goodness of SDN, where the centralized controller has the global view of the network system, we can use deterministic network coding to find the global optimum for the network system. We can assume that the size of the network is at the scale of Singapore (a medium-sized network), so the latency between the centralized controller and the switches is negligible. However, for large networks, the synchronization between routers for the deterministic network coding may be costly.

Our main contributions can be specified as follows:

1. We design a cost model to capture the network usage (e.g., network bandwidth and storage) with the aid of storage helpers for the network stakeholders; our

work is different from the work in Lun et al. [63], since their work doesn't consider the effect of helpers to the network dynamics and code constructions (we include one more set of storage constraints in the graph-theoretic region, see Equation 5.6). This extended model thus allows us to capture fog/edge computing set-ups where the network operations such as data processing and caching are essential at the network edge. Even though the cost model in our work is similar to the work in [80, 81]; their work, however, is focused on VoD system where the network devices can be divided into three layers, e.g., the layer of video servers, the layer of helpers and the layer of users where the layer of video servers and layer of helpers can be reached in one hop². For real networks, there is a layer of network backbone separating the layer of network edge devices and the layer of data centers where the intermediate network nodes in the network backbone can execute other network functionality (e.g., network coding) on the data flow and thus modify the data flow. Therefore, assuming that the distance between the layer of network edge devices and the layer of data centers is one hop is not a realistic assumption in IoT model. Moreover, their version of random network coding is performed only once at the video server. In our design, there is a clear separation between the helpers' flows and receivers' flows; this separation allows the consideration of adding the functionality of local repairable codes for the layer of distributed storage helpers to provide reliability for the cached video data. We intend to pursue this as future works by giving a detailed study (Section 5.1).

2. We emphasize on a hybrid approach, rather than depending solely on either a random version or a deterministic version in the network code construction for the feasible solution due to the reasons, as explained above. See Section 5.4 for the description of the hybrid model. To evaluate the proposed hybrid approach, we create a simulation platform to test the performance of both network coding paradigms for video transmission over SDN network by using ns-3, kodo and MPEG-4 (Section 6) and verify our claims for the hybrid approach (Section 6.2.2).

We provide a potential application for our proposed system having a set of stor-

²In other words, the network backbone is "removed".

age helpers located at the network edge as follows. During traffic peak hours, a large number of vehicles and pedestrians are entering into the transportation system. Thus, one can expect a high volume of traffic information (including visual information) is generated from the traffic devices connected to the network edge; the generated data is transmitted to the traffic center to monitor and control the transportation system. Also, the volume of generated data from the cameras and sensors of the surveillance system during the traffic peak hours is high relative to the volume of data generated during other periods. Therefore, we can assume that the peak hours of a video surveillance system in ITS is equivalent to the peak hours of a transportation system. On the other hand, fewer vehicles are traveling on the road at night (off-peak hours). Moreover, various intelligent motion and object detection methods[3] are used in the domain of video surveillance system, so when there is no object being detected, no data or data with reduced generating rate will be transmitted from cameras and sensors to edge storage nodes.

Since the availability of network bandwidth is in contention during the peak hours and there are abundant bandwidth resources during the off-peak hours (e.g., at night), we can send the crucial data to the data centers that is sufficient for global control while keeping the not-so-crucial data at network edge helpers temporarily. During off-peak hours, the network edge helpers can transmit the cached data to the data centers. More precisely, we can consider the multilayer structure of video data where the base layer (layer 1) is crucial in displaying the video and additional layers (layer 2 and above) are the enhancement of video quality. We may transmit only the base layer to data center while storing the additional layers temporarily at local storage helpers during the peak hours; we then transfer the additional layers during the off-peak hours for backup. This thus can reduce the workload over the network and increase the network efficiency.

Chapter 4

A Distributed Multilevel Storage Infrastructure (DMSI) for Visual Surveillance in ITS

In Section 4.1 and Section 4.2, we formulate goals, core functionalities and challenges for DMSI for visual surveillance in ITS. While many of them are technical in nature, some are trade-offs that depend on preference, for example, the trade-off between storage capacity and network bandwidth. Note that all the issues being considered in this chapter mainly fall under the level 3 to level 6 (from fog computing layer, data accumulation layer, data abstraction layer, to application layer) of the Cisco's Internet of Things Reference Model (See Figure 4.1). We also propose a linear optimization model with graph-theoretic constraints relying upon network coding methods in Section 5.1. The model readily extends to include any set of data sources, not restricted to video data, since the data sources are formulated as some set of random processes. However, the problem we address is pronounced when one wants to deal with the high volume and velocity of video surveillance data, and that is why we restrict the discussion to video data generated by cameras.

4.1 Design Goals

Realizing DMSI for ITS services should be able to achieve the following goals by taking the advantages of underlying fog/cloud. While fog nodes provide localization

Figure 4.1: Internet of Things Reference Model. Adapted from [39].

and local functionalities, the cloud provides global centralized management.

- *Support for Intelligent Surveillance Edge Devices and DDITS:* The local intelligent traffic devices may use the stored data on local attached storage for local traffic control and management, while the traffic control center may use the stored data on data center to make global prediction and monitoring for the whole transportation system. DMSI should provide efficient data access to the ITS stakeholders. Furthermore, transport users and drivers can connect to edge devices or data center for traffic information and services.
- *Location-aware Resource Management:* To enable efficient execution of ITS services, DMSI must consider network bandwidth along with the computation and storage capacities of edge devices. As a result, resource management decisions must optimize on computation time as well as data movement and storage costs.
- *Reliability and Fault Tolerance:* The edge devices process generated data by suitable coding techniques and then distribute encoded data across edge devices and to the data center for processing, storing and archiving. The suitability for coding might be determined by the scalable structure of video, network topology and network dynamics, including available bandwidths and storage capacities, and timeliness. Edge devices can collaborate with each other for data retrieval.

Figure 4.2: A Proposed Architecture of Joint Source, Network Coding Theory and Linear Optimization-based Distributed Multilevel Storage System over OpenFlow Network for Visual Surveillance in ITS. To Appear in IEEE Internet Computing Magazine [106].

4.2 Underlying Functionality Requirements

How can we efficiently distribute the video received from intelligent traffic surveillance devices across DMSI which is formed by the interconnection of intelligent storage nodes at the edge, intermediate nodes (either switches or intelligent storage nodes), and data center by making use of a centralized controller for data reliability and protection, and network efficiency?

As we mentioned in Chapter 1, we advocate a joint source, network coding and linear optimization theoretic approach over DMSI based on OpenFlow framework in which the network coding [56] can be implemented in a centralized manner. Figure 4.2 shows a potential joint OpenFlow controller and DMSI model for ITS, which illustrates the integration of fog/cloud, software-related concepts and multiple local intelligent surveillance systems for ITS into a single framework. In the figure, level 1 corresponds to data center, while the additional levels correspond to edge clouds. Moreover, there is a centralized server running OpenFlow controller (residing in the control plane) which can exert control logic through the control flow (red arrows in the figure) on the network devices since it has the global view of the network switches and storage (residing in the data plane) by collecting the feedback (blue arrows in the figure) from the network through the control loop. The controller is possibly logically centralized but physically distributed over the network, which functions as a *network operating system*. The communication between control plane and data plane can be implemented by using TCP protocol to assure the control and feedback delivery while the data transmission in data plane can be implemented by using standard streaming protocols. The salient point is, we can implement a linear optimization [63, 64] or control model [105] (e.g., model predictive control (MPC), optimal control, etc) with some cost metrics (e.g., minimizing the control effort, minimizing the network latency or maximizing the system efficiency) in the centralized controller to monitor and control the centralized network coding operations. We will elaborate the model functionalities and the generic workflow of centralized controller as follows.

4.2.1 Storage and Coding Scheme

The purpose of a distributed storage system (DSS) is to store information reliably over a large collection of storage nodes which are individually prone to failure. Era-

sure codes have been deployed in such DSSs to achieve reliability while maintaining high storage efficiency [41, 42, 43]. Given an object of size M , a DSS deployed with a classical erasure code (n, k) , such as Reed-Solomon codes [42], first divides the object into a set of k blocks each with size M/k , and then uses the underlying erasure code to map these blocks to another set of n blocks each with the same size M/k and disperse those n blocks across n distinct storage nodes in the system. The encoding operation is a matrix-vector multiplication that costs $O(nk)$ multiplicative computations while the decoding operation can be done by using Gaussian Elimination where the computational complexity is $O(k^3)$. Among the tremendous number of erasure codes, codes that possess the maximum distance separable (MDS) [41] property receive particular attention. For any DSS deploying with MDS codes, any k distinct encoded blocks from k distinct storage nodes out of the set of n storage nodes are sufficient to reconstruct the original file and so these codes are optimum in the reliability/storage overhead trade-off. Thus, the DSS is able to tolerate up to $n - k$ failed nodes. A famous example is the deployment of HDFS/RAID in facebook’s data warehouses [43].

How should we design a coding scheme that is tailored for DMSI for ITS, so that the video data can be stored more reliably? In our context, this is in fact a joint source and network coding scheme, which can function cooperatively with the aid of OpenFlow controller. To design such a scheme, we can consider the scalability of video structure [44, 52], which is a video coding method removing parts of the original video bit stream so that the resulting substream can be reconstructed with lower spatial or temporal quality relative to the original stream. In this article we consider the scalable video structures and the corresponding methods for handling those structures from two aspects: (a) multilayer structure, and (b) frame structure.

In the multilayer structure of video stream (MRC) [46, 47], every segment of the stream can be divided into several layers where the base layer (layer 1) is required to display the video, and additional layers (layer 2 and above) are the enhancement of video quality. The combination of layer 2 and layer 3 without the base layer is useless. Recent developments in Video-On-Demand (VoD) systems has led to the creation of new variants of MRC, which rely on linear network coding, for the multilayer structure [80, 81, 52]. Data packets from different layers (interlayer) or same layer (intralayer) are combined and then distributed across the multilevel networked

storage nodes of VoD system for storing and archiving; those nodes can collaborate together for data retrieval. Since there is an analogy between our proposed DMSI and the storage system in VoD, it is tempting to borrow the idea from MRC and apply in the realm of ITS, by considering the inherent characteristics of video (such as video structure).

In addition, we can zoom into the inner frame structure of video in which video stream consists of many frames (e.g., 24 – 48 fps for video). We can divide the video into many equally important substreams with less frames, and encode those substreams in a way that the decoding quality is acceptable by using a subset of those substreams. By using the MDC [48], it is feasible to decode the media stream by using any substream (the description) and improve the quality when multiple substreams are received in parallel [48, 52]. For example, we can transmit only an acceptable set of substreams to data center and retain the remaining substreams at edge storage nodes (which may be sent to the back-end data center opportunistically), thus saving network bandwidth.

4.2.2 Network Bandwidth Allocation and Content Placement

For each video segment, which packets of which layers should a local storage node (or data center) store to maximize the system performance? Which packets, and to which neighborhood of storage nodes, should each local storage node (or data center) forward its coded content? These are some questions we have to answer when designing a multi-level DSS for ITS. Moreover, any network bandwidth allocation scheme must consider the workload from tens of thousands of nodes that may overwhelm the network bandwidth of a data center. Furthermore, since the groups of local storage nodes are overlapped and interconnected with each other (may be separated by several street blocks), and connected to data center (connecting by citywide storage networks), we have to consider the neighborhood selection problem so that the data distribution of the designed coding scheme for the whole system is judicious.

We have seen many ongoing explorations that apply network coding in VoD studies [52] for more efficient video streaming. In fact, we can make use of the

results in VoD domain and extend to our case by creating a network model for efficient streaming of the video data created from multiple surveillance cameras and multicasting, broadcasting or unicasting to data centers on top of a virtualized network. For example, SDM²Cast [35] enables multicast video streaming on top of an overlay network.

We want to point out that there is a fundamental difference between the data flow and setup in the VoD system and the case of ITS, where the data center and intermediate nodes in VoD are deployed to serve user nodes (streaming), while in our case, the collected video data are pushed to intermediate nodes with storage capability and data centers for storing, archiving or further processing (in the reverse direction, when compared to VoD system), and the edge nodes may pull data from the data center for local processing. Furthermore, the sink nodes in VoD models are the subscribers of video service, while in our case, the data center is the “only” subscriber of video (where the cameras are now acting as sources of information). Of course we can consider the case where the edge nodes pull data from the data center, which is similar to VoD models.

4.2.3 Fine-grained Control for Reliability

Edge nodes have less resources (such as computing power, storage capacity and RAM) than data centers, but the goal of fog computing is to relocate the workload from data centers to the edges to reduce the load at a data center by local processing. Less important traffic video information is cached on the storage helpers during the peak period, so that the amount of data to be transferred through the network can be reduced. The cached information can be transmitted to the data center for further processing during offpeak period. As mentioned, we can apply linear optimization [63, 64] or control theoretic approach [105] together with network coding to achieve the goal and constraints. So, the immediate question is, how to design a control model which can make use of the feedback from networked control systems to fine-tune the system through the control flow? We illustrate a generic workflow for the proposed architecture as follows.

1. Based on some predetermined sampling period, the OpenFlow controller collects the feedback from the network (blue flow in Figure 4.2). The controller

now has an approximate global view of the network.

2. The controller computes an optimal (or greedy) network coding tree with a set of accompanying network coding coefficients, based on some control laws and cost metrics, subject to some system constraints, by using the collected feedback. This can be divided into several steps.
 - First, define some set of vector-valued metric functions $g : G \mapsto \mathbb{R}^n$ that map the set of possible network topologies G to the set of real vectors $\mathbb{R}^n$. These metric functions will be used in the control model of the proposed architecture.
 - Second, compute an optimal (or greedy) parameter set based on the control model with the suitable metrics and corresponding constraints by feedback.
 - Third, use the optimal point found in the previous step to determine the best suitable network topology for centralized network coding.
 - Fourth, use a deterministic algorithm, e.g., [56] to generate the local coding coefficients for the network topology found in the previous step.
3. Then, the controller pushes the coding tree and coefficients to the network and configures the setup (red flow in Figure 4.2).
4. Finally, the switches (with reasonable computation power) and local storage execute the network coding.

In fact, the line of research to address the issues of the second step above is known as aggregate utility maximization problem and capacity region analysis. There are several works to address the problem by using network coding [71, 52], rather than the conventional method of using routing. In particular, [71] studied the achievable capacity region of some special cases of P2P topology by using network coding and showed that those cases can be reduced to routing. For the single source multicasting over general topology, it is known that the problem of determining the capacity region and the problem of constructing the coding operation can be solved independently [63, 53, 56]. However, none of those works provide a full solution for multisource multicasting over general network topology, and suboptimal and heuristic approaches are used in practice [63, 64]. Moreover, those network models are

assumed to be static in which the available network resources (network capacities) are fixed over time, but this is not the case in real-life networks, which are dynamic where the available network resources are changing from time to time.

Chapter 5

Network Coding Models for ITS

By considering several design aspects of the previous chapter (e.g., Storage and Coding Scheme, Network Bandwidth Allocation and Content Placement and Fine-grained Control for Reliability, Section 4.2), we formulate the problem using the following model in Section 5.1. Specifically, our model aims to address the later two design goals, which in turn are expected to facilitate the first one mentioned in Section 4.1. In the last section of this chapter, we will talk about our hybrid approach and its formulation. The simulation results of the hybrid approach will be given and analyzed in Section 6.2.2.

5.1 System Model and Problem Formulation

Given a directed acyclic network $G = (V, E)$ with a set of sources $s \in S \subseteq V$ to model the set of cameras and a set of receivers $t \in T \subseteq V$ to model the set of data centers where $S \cap T = \emptyset$. In other words, we want to multicast video data from a set of cameras to data centers connected by the network. We denote the set of edge capacities of G as $C = \{c_{ij} : (i, j) \in E\}$. Every source s has M_s processes $X_{1_s}, X_{2_s}, \dots, X_{M_s}$ with rates $R_{1_s}, R_{2_s}, \dots, R_{M_s}$, respectively; this is to model the multilayer structure of videos generated at cameras. For convenience, we just use the index of the source process to denote the process itself, that is, we denote 1_s to be X_{1_s} , 2_s to be X_{2_s} , etc. Let D be the collection of all source processes, that is, $D = \bigcup_{s \in S} \{1_s, 2_s, \dots, M_s\}$. Each receiver $t \in T$ requests a subset of the source processes that are generated from some subset of source nodes, which we specify as $D(t) \subset D$. We assume there are some intermediate nodes $h \in V \setminus S \cup T$, which have

the storage capability to buffer information flow and denote this collection of nodes as H . We call $h \in H$ as *helper* and denote the available storage capacity of h as B_h . This setting embodies the edge nodes and network switches attached with storage devices in fog computing.

We denote

$$\Theta = D \setminus \bigcup_{t \in T} D(t), \quad (5.1)$$

which represents the set of source processes that are not requested by any receiver $t \in T$ and use $D(h) \subset \Theta$ to represent the set of source processes going to be stored at $h \in H$.

For any node i , let $T(i)$ denote the receivers that are accessible from i , i.e.,

$$T(i) = \{t \in T \mid \exists \text{ a directed path to } t \text{ from } i \text{ or } t = i\}, \quad (5.2)$$

and let $\Gamma(i)$ denote a set partition of $\bigcup_{t \in T} D(t)$ that represents the sources that can be mixed on links going into i with destinations $t \in T$, i.e.,

$$\Gamma(i) = \left\{ \bigcap_{t \in T(i)} A(t) \mid A(t) = D(t) \text{ or } A(t) = \left(\bigcup_{t \in T} D(t) \right) \setminus D(t) \right\} \setminus \{\emptyset\}. \quad (5.3)$$

In other words, $\Gamma(i)$ is the set of atoms of the algebra generated by $\{D(t)\}_{t \in T(i)}$. For a given atom¹ $\gamma \in \Gamma(i)$, the receivers that receive a source process in γ through an edge (j, i) either receive all the source processes in γ or none at all. Hence the source processes in γ can be mixed on the edge (j, i) as the receivers that receive the mixture will also receive the source processes (or mixtures thereof) necessary for decoding. Similarly, for any node i , let $H(i)$ denote the helpers that are accessible from i , i.e.,

$$H(i) = \{h \in H \mid \exists \text{ a directed path to } h \text{ from } i \text{ or } h = i\}, \quad (5.4)$$

and let $\Psi(i)$ denote a set partition of Θ that represents the sources that can be mixed on links going into i with destinations $h \in H$, i.e.,

$$\Psi(i) = \left\{ \bigcap_{h \in H(i)} A(h) \mid A(h) = D(h) \text{ or } A(t) = \Theta \setminus D(h) \right\}. \quad (5.5)$$

¹A nonempty set γ is called an atom of an algebra Γ if $\gamma \in \Gamma$ and for all $\psi \in \Gamma$, either $\gamma \subset \psi$ or $\gamma \cap \psi = \emptyset$.

Let a_{ij} be the cost per unit bandwidth for the edge (i, j) and b_h be the cost per unit storage for the helper h . We refer d as the destination node, i.e., $d = t$ or $d = h$. Denote $f_{ij}^{(d, m_s)}$ where $1_s \leq m_s \leq M_s$ and $s \in S$ to be the allocated bandwidth across the edge (i, j) when the connection (d, m_s) is established. In this multisource multicasting model, a natural goal is to minimize the aggregate cost with the aid of intermediate nodes with storage capability, subject to the constraints of bandwidth usage z_{ij} and storage usage r_h , i.e.,

$$\begin{aligned}
& \min_{z, r} \quad \sum_{(i, j) \in E} a_{ij} z_{ij} + \sum_{h \in H} b_h r_h, \\
& \text{s.t.} \quad c_{ij} \geq z_{ij} = \sum_{\gamma \in \Gamma(j)} y_{ij}^\gamma + \sum_{\psi \in \Psi(j)} w_{ij}^\psi, \quad \forall (i, j) \in E, \\
& \quad y_{ij}^\gamma \geq \sum_{m_s \in \gamma} f_{ij}^{(t, m_s)}, \quad \forall (i, j) \in E, t \in T, \gamma \in \Gamma(j), \\
& \quad w_{ij}^\psi \geq \sum_{m_s \in \psi} f_{ij}^{(h, m_s)}, \quad \forall (i, j) \in E, h \in H, \psi \in \Psi(j), \\
& \quad \sum_{\{j | (i, j) \in E\}} f_{ij}^{(d, m_s)} - \sum_{\{j | (j, i) \in E\}} f_{ji}^{(d, m_s)} = \sigma_i^{(d, m_s)}, \\
& \quad \quad \quad \forall i \in V, d \in T \cup H, m_s \in D(d), \\
& \quad r_h = \sum_{m_s \in D(h)} \sigma_i^{(h, m_s)} \leq \min\{U_h, \frac{U}{\tau}\}, \quad \forall h \in H, \\
& \quad f_{ij}^{(d, m_s)} \geq 0, \quad \forall (i, j) \in E, d \in T \cup H, m_s \in D(d),
\end{aligned} \tag{5.6}$$

where

$$\sigma_i^{(d, m_s)} = \begin{cases} R_{m_s} & \text{if } i = s \in S \text{ and } m_s \in D(d); \\ -R_{m_s} & \text{if } i = t \text{ and } m_s \in D(t); \\ -R_{m_s} & \text{if } i = h \text{ and } m_s \in D(h); \\ 0 & \text{otherwise.} \end{cases} \tag{5.7}$$

Basically, there are two types of information flows, one is from sources to sinks $\{f_t\}$ and another is from sources to storage helpers $\{f_h\}$ (those flows are video layers not requested by the data centers and thus should be cached on helpers) where flows from each type are never to be mixed (or linearly combined) at the intermediate nodes. We only allow the mixture between flows that are under the same session [56]. In a simplified version, we want to achieve the following objective:

$$\begin{aligned}
& \min_{z,r} \quad \sum_{(i,j) \in E} a_{ij} z_{ij} + \sum_{h \in H} b_h r_h, \\
& \text{s.t.} \quad \text{the flows } \{f_t\}, \{f_h\} \text{ satisfy the edge constraints } C, \\
& \quad \text{the flows } \{f_t\}, \{f_h\} \text{ satisfy the node constraints with} \\
& \quad \text{values } \{\pm R_{ms}\} \text{ or } 0, \\
& \quad \text{the buffered video layers } \{r_h\} \text{ at helpers, which depend} \\
& \quad \text{on the flows } \{f_h\}, \text{ satisfy the storage constraints } \{B_h\}.
\end{aligned} \tag{5.8}$$

Given the feasible graph-theoretic region in Equation 5.6 (or 5.8) above, we would like to study the existence of network codes that configure multicast connections for the network model. There are 3 sets of constraints in the feasible region, where the first set captures the limitation of network bandwidth across every link in the network on the both types of flows (edge constraints), the second set represents the constraints on the summation of flows when flowing through network nodes that can be seen as the flow conservation law (node constraints), and the third set represents the amount of storage that the helper can buffer (storage constraints).

By searching an optimal solution for the linear programming (LP) model in Equation 5.6 satisfying the 3 sets of constraints, we can address the issue of network bandwidth allocation and content placement in Section 4.2.2, since the resulting solution must be optimal in term of the storage and bandwidth costs with respect to the constraints. By using the techniques from the network coding theory, we can construct a network code for the optimal solution found for the LP model and thus provide a storage and coding scheme for the video transmission over the network from cameras to data centers and buffering at the storage helpers, as mentioned in Section 4.2.1. These together take care of the second design goal.

Due to the separation of different types of flows, either from sources to sinks or from sources to helpers, we can deploy network codes [56, 57] for the first type of flows to provide reliability and protection for timely video transmission, and apply regenerating codes [103] or locally repairable codes [104] for the second type of flows to provide reliability and local repairability for the unrequested video layers that are going to be stored on distributed helpers over the SDN network. While regenerating codes are variants of network codes that provide repair property for DSS, locally repairable codes are a new class of codes that allow to locally repair a lost data

fragment and retrieve back the data fragment without decoding the whole data. The combination of SDN controller with the LP model (where we run the LP model as a service atop network operating system) allows us to achieve fine grained control of video data (Section 4.2.3) over the SDN network since we now have the global view of the network connecting edge devices, helpers and data centers, and therefore we can compute the optimal or nearly optimal network codes or regenerating codes for the video transmission and buffering. These take care of the third design goal.

In general, being a feasible solution satisfying the constraints and the existence of a network code for multicast connections for the feasible solution is not equivalent. This is important and worth exploring, since the set of intermediate nodes with storage capacities in our traffic surveillance model captures the roles of fog computing, in which edge nodes are able to store and process data locally. Moreover, the idea of putting storage devices close to intermediate nodes in the network and applying network coding will change the network dynamics, and thus this shall open opportunities for new coding solutions for the setup.

5.2 Linear-Information-Flows Algorithm

In this section we provide our algorithm to construct a network code for each feasible solution in the feasible region of Equation 5.6, and we give the claims and the proofs in the next section.

We consider the special case by assuming that the rates are unity (one symbol per unit time), i.e., $R_{m_s} = 1$ for all m_s where $1_s \leq m_s \leq M_s$ and $s \in S$. Thus, there are $|D(t)|$ unit flow paths flowing from sources $s \in S$ to each receiver $t \in T$ and $|D(h)|$ unit flow paths flowing from sources $s \in S$ to each helper node $h \in H$. Our algorithm and proof follow a similar construction as [56] and [63]. The main idea here is we want to maintain a collection B_d of global coding vectors for each destination $d \in T \cup H$, either a helper or a receiver, such that this collection of coding vectors spans the information subspace be requested by each of the destination when stepping through each node in the network by upstream-to-downstream manner. This is equivalent to $\text{span}(B_d) = \text{span}(\bigcup_{m \in D(d)} B(s', s)_m)$ for each destination d where we denote $(s', s)_m$ as imaginary edge connecting imaginary source s' and s (the precise definition is given in the algorithm).

Algorithm 1: A LIF for DMSI in ITS

```

1: function LIF( $G = (V, E, C), S, T, H$ )
2:   #Preparation#
3:   Run flow assignment algorithm  $flow\_assignment(G = (V, E, C), S, T, H)$  to
   compute the set of unit flow paths  $\{f^{(t,m)}\}$  from sources to receivers and the
   set of unit flow paths  $\{f^{(h,m)}\}$  from sources to helpers w.r.t. the constraints in
   Equation 5.6
4:   Let  $F$  be a finite field such that  $|F| \geq \max(|T|, |H|)$ 
5:   Add  $|S|$  many imaginary nodes  $s'$  into  $V$  and pair each  $s'$  to one  $s \in S$  and
   denote the collection of imaginary nodes as  $S'$ 
6:   For each pair  $(s', s)$ , connect  $M_s$  parallel edges  $(s', s)_{1_s}, (s', s)_{2_s}, \dots, (s', s)_{M_s}$ 
   from  $s'$  to  $s$ 
7:
8:   #Initialization#
9:   for  $m = 1 \rightarrow |D|$  do
10:     $b((s', s)_m) := [0^{m-1}, 1, 0^{|D|-m}]$ 
11:   for all receivers  $t \in T$  do
12:     $C_t := \{(s', s)_m\}_{m \in D(t)}$  and
13:     $B_t := \{b((s', s)_m)\}_{m \in D(t)}$ 
14:   for all helpers  $h \in H$  do
15:     $C_h := \{(s', s)_m\}_{m \in D(h)}$  and
16:     $B_h := \{b((s', s)_m)\}_{m \in D(h)}$ 
17:
18:   #InductiveStep#
19:   for all vertex  $i \in V \setminus S'$  in topological order (by longest directed path) do
20:     for all outgoing edges  $(i, j)$  of  $i$  do
21:       Define  $B(i, j) := \bigcup_{\gamma \in \Gamma(j)} B_\gamma(i, j) \cup \bigcup_{\psi \in \Psi(j)} B_\psi(i, j)$ 
22:       Run  $receiver\_algorithm((i, j), \Gamma(j), T)$ 
23:       Run  $helper\_algorithm((i, j), \Psi(j), H)$ 
24:   return (a set of unit flow paths  $\{f^{(t,m)}\} \cup \{f^{(h,m)}\}$ , a finite field  $F$ , local
   coding coefficients)

```

Inside the inner for loop of the inductive step of the main algorithm, there are

two subroutines, which are helper algorithm (Algorithm 2) and receiver algorithm (Algorithm 3). The receiver algorithm captures the streaming aspect of the video through the network, while the helper algorithm captures the buffering property of the helpers. For each edge (i, j) , we have

$$B(i, j) := \bigcup_{\gamma \in \Gamma(j)} B_\gamma(i, j) \cup \bigcup_{\psi \in \Psi(j)} B_\psi(i, j) \quad (5.9)$$

where $B_\gamma(i, j)$'s and $B_\psi(i, j)$'s are disjoint and if there exists flow flowing through the edge (i, j) for the connection (d, m) , i.e., $f_{ij}^{(d, m)} = 1$, it is associated to global coding vector in $B_\gamma(i, j)$ (if $t \in T$) for the unique γ such that $m \in \gamma$ or global coding vector in $B_\psi(i, j)$ (if $h \in H$) for the unique ψ such that $m \in \psi$.

Note that we handle the set of receivers and the set of helpers by different flows which are never overlapped ($z_{ij} = \sum_{\gamma \in \Gamma(j)} y_{ij}^\gamma + \sum_{\psi \in \Psi(j)} w_{ij}^\psi, \forall (i, j) \in E$ in Equation 5.6), so we can separate the body of the inner for loop into two subroutines. Moreover, due to the separation of flows, we may replace the code construction for the helper nodes in this report by regenerating codes [103] or locally repairable codes [104] to capture the distributed property of those components; however, this possibility is not explored in this report.

Algorithm 2: Receiver Algorithm

- 1: **function** RECEIVER($((i, j), \Gamma(j), T)$)
- 2: **for all** $\gamma \in \Gamma(j)$ **do**
- 3: Let $t^* := \arg \max_{t \in T} f_{ij}^{(t, \gamma)}$ where $f_{ij}^{(t, \gamma)} = \sum_{m \in \gamma} f_{ij}^{(t, m)}$
- 4: Define $S_T := \{t \in T | f_{ij}^{(t, \gamma)} > 0\}$
- 5: Construct $f_{ij}^{(t^*, \gamma)}$ many imaginary unit edges $(i, j)_m$ for $m \in \gamma$ along (i, j)
if $f_{ij}^{(t, m)} = 1$
- 6: Choose a set of global coding vectors $B_\gamma(i, j)$ such that for every $b((i, j)_m) \in B_\gamma(i, j)$, $B_t \setminus \{b(f_{p, ij}^{(t, m)})\} \cup \{b((i, j)_m)\}$ is linearly independent for every $t \in S_T$ w.r.t. t^* where $f_{p, ij}^{(t, m)}$ denoting the predecessor edge of $(i, j)_m$ along the unit flow path $f^{(t, m)}$
- 7: **for all** $t \in S_T$ **do**
- 8: Define $(i, j)_t := \{(i, j)_m | f_{ij}^{(t, m)} = 1\}$
- 9: **for all** $(i, j)_m \in (i, j)_t$ **do**
- 10: $C'_t := C_t \setminus \{f_{p, ij}^{(t, m)}\} \cup \{(i, j)_m\}$


```

11:       $B'_t := B_t \setminus \{b(f_{p,ij}^{t,m})\} \cup \{b((i,j)_m)\}$ 
12:       $a'_t := \frac{a(f_{p,ij}^{(t,m)})}{b((i,j)_m) \cdot b(f_{p,ij}^{(t,m)})}$ 
13:      for all  $c \in C_t \setminus \{f_{p,ij}^{(t,m)}\}$  do
14:           $a'_t = a_t(c) - (b((i,j)_m) \cdot b(f_{p,ij}^{(t,m)}))a'_t$ 
15:       $(C_t, B_t, a_t) := (C'_t, B'_t, a'_t)$ 

```

Algorithm 3: Helper Algorithm

```

1: function HELPER( $((i,j), \Psi(j), H)$ )
2:   for all  $\psi \in \Psi(j)$  do
3:     Let  $h^* := \arg \max_{h \in H} f_{ij}^{(h,\psi)}$  where  $f_{ij}^{(h,\psi)} = \sum_{m \in \psi} f_{ij}^{(h,m)}$ 
4:     Define  $S_H := \{h \in H \mid f_{ij}^{(h,\psi)} > 0\}$ 
5:     Construct  $f_{ij}^{(h^*,\psi)}$  many imaginary unit edges  $(i,j)_m$  for  $m \in \psi$  along  $(i,j)$ 
       if  $f_{ij}^{(h,m)} = 1$ 
6:       Choose a set of global coding vectors  $B_\psi(i,j)$  such that for every
          $b((i,j)_m) \in B_\psi(i,j)$ ,  $B_h \setminus \{b(f_{p,ij}^{(h,m)})\} \cup \{b((i,j)_m)\}$  is linearly independent for
         every  $h \in S_H$  w.r.t  $h^*$  where  $f_{p,ij}^{(h,m)}$  denoting the predecessor edge of  $(i,j)_m$ 
         along the unit flow path  $f^{(h,m)}$ 
7:       for all  $h \in S_H$  do
8:         Define  $(i,j)_h := \{(i,j)_m \mid f_{ij}^{(h,m)} = 1\}$ 
9:         for all  $(i,j)_m \in (i,j)_h$  do
10:             $C'_h := C_h \setminus \{f_{p,ij}^{(h,m)}\} \cup \{(i,j)_m\}$ 
11:             $B'_h := B_h \setminus \{b(f_{p,ij}^{h,m})\} \cup \{b((i,j)_m)\}$ 
12:             $a'_h := \frac{a(f_{p,ij}^{(t,m)})}{b((i,j)_m) \cdot b(f_{p,ij}^{(h,m)})}$ 
13:            for all  $c \in C_h \setminus \{f_{p,ij}^{(h,m)}\}$  do
14:                 $a'_h = a_t(c) - (b((i,j)_m) \cdot b(f_{p,ij}^{(h,m)}))a'_h$ 
15:             $(C_h, B_h, a_h) := (C'_h, B'_h, a'_h)$ 

```

5.3 Claims and Proofs

We provide some preliminary lemmas and then give the main result of this report. After that, we prove the correctness of the algorithm and give the complexity analysis.

The following lemma [56, Lemma 5] provides a way to test for the linear independence on subspace.

Lemma 1 *Consider a basis B of $F^{|D|}$ and vectors $b \in B$, $a \in F^{|D|}$ such that for every $b' \in B$, we have $b' \cdot a = \delta_{b,b'}$. Then any vector $x \in F^{|D|}$ is linearly independent on $B \setminus \{b\}$ if and only if $x \cdot a \neq 0$.*

Proof Suppose that $x = \sum_{b' \in B} x(b')b'$ be the unique representation in the basis B . Then we have

$$x \cdot a = \left(\sum_{b' \in B} x(b')b' \right) \cdot a = \sum_{b' \in B} x(b')(b' \cdot a) = \sum_{b' \in B} x(b')(\delta_{b,b'}) = x(b). \quad (5.10)$$

Therefore, $x \in F^{|D|}$ is linearly dependent on $B \setminus \{b\}$ if and only if $x \cdot a = x(b) = 0$. This is equivalent to say that $x \in F^{|D|}$ is linearly independent on $B \setminus \{b\}$ if and only if $x \cdot a \neq 0$. ■

In Algorithm 2, how to choose a collection of global coding vectors $B_\gamma(i, j)$ such that for every $b((i, j)_m) \in B_\gamma(i, j)$, $B_t \setminus \{b(f_{p,ij}^{(t,m)})\}$ is linearly independent for every receiver $t \in S_T$ where $f_{p,ij}^{(t,m)}$ denoting the predecessor edge of $(i, j)_m$ along the path $f^{(t,m)}$? Similarly, how to do this for the collection $B_\psi(i, j)$ in Algorithm 3? The following lemma provides an answer to this.

Lemma 2 *Let $n \leq |F|$. Consider pairs $(x_k, y_k) \in F^{|D|} \times F^{|D|}$ with $x_k \cdot y_k \neq 0$ for $1 \leq k \leq n$. There exists a linear combination u of $x_1, x_2, \dots, x_n$ such that $u \cdot y_k \neq 0$ for $1 \leq k \leq n$.*

Proof We prove it by induction on n such that for all k, j with $1 \leq j \leq k \leq n$, $u_k \cdot y_j \neq 0$ and put $u = u_n$. For the initial step $n = 1$, we put $u = u_1 := x_1$ and so $u \cdot y_1 \neq 0$.

Now let k be such that $1 \leq k \leq n - 1$. There are two cases, either $u_k \cdot y_{k+1} \neq 0$ or $u_k \cdot y_{k+1} = 0$. If $u_k \cdot y_{k+1} \neq 0$ then we done and put $u_{k+1} := u_k$. Else, we have $(\alpha u_k + x_{k+1}) \cdot y_{k+1} \neq 0$ for every $\alpha \in F$ since $x_{k+1} \cdot y_{k+1} \neq 0$. For every j where $1 \leq j \leq k$, we have the condition

$$(\alpha u_k + x_{k+1}) \cdot y_j = 0 \text{ if and only if } \alpha = \alpha_j := -\frac{x_{k+1} \cdot y_j}{u_k \cdot y_j}. \quad (5.11)$$

So for each j there is exactly one "bad" α_j and since $|F| \geq n > k$, we are able to choose "good" α and define $u_{k+1} := \alpha u_k + x_{k+1}$. By construction, we have $u_{k+1} \cdot y_j \neq 0$ for $1 \leq j \leq k+1$. ■

Now we give the main result of this report and its proof.

Theorem 2 *Suppose z is a feasible solution in the feasible region of Equation 5.6. Then Algorithm 1 constructs a network code that sets up multicast connections for the source processes $m_s \in D$ at a rate $R_{m_s} = 1$ from sources $s \in S$ to the destinations in the set $\{d \in T \cup H | m_s \in D(d)\}$ with the flow rate z_{ij} at each edge (i, j) .*

Proof Data (video information) received at each destination d (e.g., receiver and helper) is decodable if and only if the set of global coding vectors B_d associated to unit flow paths $\{f^{(d,m)}\}$ transferring to d spans the information space $\text{span}(\bigcup_{m \in D(d)} b((s', s)_m))$ requested by d . This is equivalent to

$$\text{span}(\bigcup_{m \in D(d)} b((s', s)_m)) = \text{span}(B_d) \subset \text{span}(\bigcup_{\{i | (i,d) \in E\}} B(i, d)). \quad (5.12)$$

Therefore, we want to show that the equality in Equation 5.12 holds at node d . To achieve the goal, our algorithm maintains the following two invariants when stepping through the nodes in the network by upstream-to-downstream order:

Invariant 1: For each destination $d \in T \cup H$,

$$\text{span}(B_d) = \text{span}(\bigcup_{m \in D(d)} b((s', s)_m)). \quad (5.13)$$

Invariant 2: For every edge (i, j) ,

$$B_\gamma(i, j) \subset \text{span}(\bigcup_{m \in \gamma} b((s', s)_m)) \text{ and } B_\psi(i, j) \subset \text{span}(\bigcup_{m \in \psi} b((s', s)_m)) \quad (5.14)$$

Before stepping through the nodes, consider the sets of global coding vectors $B((s', s)_m) = \{b((s', s)_m)\}$ where $m \in D$. Then for each d , it is evident that (defined in Algorithm 1)

$$B_d := \{b((s', s)_m)\}_{m \in D(d)} = \bigcup_{m \in D(d)} b((s', s)_m) \quad (5.15)$$

Also, we have

$$B_\gamma((s', s)_m) \subset \text{span}(\bigcup_{m \in \gamma} b((s', s)_m)) \quad (5.16)$$

if it is destined to $t \in T$ or

$$B_\psi((s', s)_m) \subset \text{span}(\bigcup_{m \in \psi} b((s', s)_m)) \quad (5.17)$$

if it is destined to $h \in H$ since the flows of source processes to each type of destinations (either h or t) are not overlapped.

Now, assume that both properties are thus far satisfied and consider a vertex i and an outgoing edge (i, j) . We have the following claim, where the proof is given after the theorem:

Claim 1: If $|F| \geq \max\{|T|, |H|\}$, then we have $\dim(B_d) = |D(d)|$ for every destination d .

Also, for the collection of receivers, we have $T(i) \supset T(j)$ since j is ordered after i and there may exist other directed paths passing through other edges (i, j') where $j \neq j'$, and suppose that $m \in \gamma \in \Gamma(j)$. If $\gamma' \in \Gamma(i)$ is the unique element such that $m \in \gamma'$, then $\gamma' \subset \gamma$. Therefore, by invariant 2,

$$\begin{aligned} B_{\gamma'}(j', i) \cup \dots \cup B_{\gamma^n}(j^n, i) &\subset \text{span}(\bigcup_{m \in \gamma'} b((s', s)_m) \cup \dots \cup \bigcup_{m \in \gamma^n} b((s', s)_m)) \\ &= \text{span}(\bigcup_{m \in \gamma} b((s', s)_m)) \end{aligned} \quad (5.18)$$

where $\gamma = \gamma' \cup \dots \cup \gamma^n$ for some integer n . It follows that for every $b \in B_\gamma(i, j)$

$$b \in \text{span}(B_{\gamma'}(j', i) \cup \dots \cup B_{\gamma^n}(j^n, i)) \subset \text{span}(\bigcup_{m \in \gamma} b((s', s)_m)). \quad (5.19)$$

Hence, we get $B_\gamma(i, j) \subset \text{span}(\bigcup_{m \in \gamma} b((s', s)_m))$. But, for $t \in S_T$ (defined in Algorithm 2),

$$\text{span}(\bigcup_{m \in \gamma} b((s', s)_m)) \subset \text{span}(\bigcup_{m \in D(t)} b((s', s)_m)) \quad (5.20)$$

as $\gamma \subset D(t)$. It follows that

$$\text{span}(B_t) \subset \text{span}(\bigcup_{m \in D(t)} b((s', s)_m)) \quad (5.21)$$

and since $\dim(B_t) = |D(t)|$ by claim 1, the equality holds for Equation 5.21. We can use the same argument for the collection of helpers.

Since z is a feasible solution, we have $z_{ij} \geq |B(i, j)|$ at every edge (i, j) . This completes the proof. ■

5.4 Hybrid Network Coding

In this section, we talk about our innovative approach integrating both network coding paradigms for ITS over the Internet.

Since the Internet can be modeled as a scale-free network [101, 102] and video data is transmitted across the internet from one end to the other, we can assume that our underlying network models can be described by Barabasi-Albert model [102] or Waxman model [101] (See Section 6.1.1 for further analysis). We assume that we have several clusters of nodes where each cluster operates as an autonomous system (AS, in the language of networking) and is at small to medium scales in terms of geographical distance (See Figure 5.1). This assumption is important since we want to assure that the propagation delay and latency of the studied network models are under control so that we can apply deterministic network coding on each individual cluster. These clusters are interconnected and thus the collection of clusters occupies a larger geographical region. Therefore, the data has longer traveling time or hops (higher latency) when traveling across clusters than traveling within a cluster. Moreover, the events of network change (e.g., churn or nodes joining and leaving) within a cluster are relatively rare and thus the cluster (local network) is stable. Each cluster has several micro or edge data centers to provide computing or storage resources catering the need of network users, such as storage backup for video data in our case. The camera sources can transmit the video data either to the data centers locating in the same cluster for backup to provide availability and fast recovery or to data centers locating in other clusters to assure redundancy.

The main idea of our hybrid approach, as we mentioned in the previous paragraph, is that we can apply deterministic network coding for the video data transmissions *within a cluster* to have the benefit of higher throughput gain since no linear dependency issue incurred by random network coding and lower computational cost in terms of finite field computation since a smaller field size is applied. For the data transmissions *across clusters*, we apply random network coding for the robustness to combat packet loss rates and dynamical network change, which is the downside of deterministic network coding. By this set-up, we expect to get the benefits from both network coding paradigms.

For simplicity, we apply intrasession network coding in our hybrid model for ITS. First we define some terminologies for the discussion. For the camera sources and

Figure 5.1: A Collection of Interconnected Scale-free Networks. In our hybrid approach, we apply deterministic network coding for a *within-cluster multicasting group* and apply random network coding for a *cross-cluster multicasting group*. A within-cluster multicasting group includes a source and all the collocated data centers; a cross-cluster multicasting group include a source and all the non-collocated data centers at the target cluster.

data centers located in the same cluster, we say that they are *collocated*; otherwise, they are *non-collocated*. For video transmissions within a cluster, each source and all the collocated data centers form a *within-cluster multicasting group* where all the data centers demand the same data from the source. Similarly, for video transmissions across clusters, each multicasting source and all the non-collocated data centers at the target cluster form a *cross-cluster multicasting group* where all the data centers demand the same data as well. We denote a multicasting group as $(s, t \in T_s \subset T)$ where T_s denotes all the data centers in the target cluster.

Given a clustered network $G = (V, E)$ with a set of sources $s \in S \subseteq V$ to model the set of cameras and a set of receivers $t \in T \subseteq V$ to model the set of collocated or non-collocated data centers where $S \cap T = \emptyset$. In other words, we want to multicast video data from a set of cameras to data centers connected by the clustered network. We denote the set of edge capacities of G as $C = \{c_{ij} : (i, j) \in E\}$. Every source s multicasts with rate R_s^t to sink t . Denote $z_{ij}^{(t,s)}$ to be the allocated bandwidth across the edge (i, j) when the connection (t, s) is established. Also, denote z_{ij}^s to be the *effective bandwidth* flowing through the edge (i, j) for the intrasession multicasting of a multicasting group $(s, t \in T_s \subset T)$. Each intrasession multicasting of a multicasting group $(s, t \in T_s \subset T)$ can be described by the following model:

$$\begin{aligned}
& \max \quad \chi, \\
& \text{s.t.} \quad R_s^t \geq \chi, \quad t \in T_s \subset T, \\
& \quad c_{ij} \geq z_{ij}^s, \quad \forall (i, j) \in E, \\
& \quad z_{ij}^s \geq z_{ij}^{(t,s)}, \quad \forall (i, j) \in E, \quad \forall t \in T_s \subset T, \\
& \quad \sum_{\{j|(i,j) \in E\}} z_{ij}^{(t,s)} - \sum_{\{j|(j,i) \in E\}} z_{ji}^{(t,s)} = \sigma_i^{(t,s)}, \quad \forall i \in V, \quad \forall t \in T_s \subset T, \\
& \quad z_{ij}^{(t,s)} \geq 0 \text{ and } z_{ij}^s \geq 0, \quad \forall (i, j) \in E, \quad t \in T_s \subset T,
\end{aligned} \tag{5.22}$$

where

$$\sigma_i^{(t,s)} = \begin{cases} R_s^{(t,s)} & \text{if } i = s \in S; \\ -R_s^{(t,s)} & \text{if } i = t \in T_s \subset T; \\ 0 & \text{otherwise.} \end{cases} \tag{5.23}$$

The first constraint assures that the min-cut is at least χ .

In our simulation, we assume each video source multicasts a video file to the collocated data centers and the same video file to all the non-collocated data centers of a target cluster. For example, suppose we have 4 clusters where each cluster has 50 sources, 150 intermediate nodes, and 2 sinks. For the simplicity of the simulation, we assume that the sources in the cluster 1 transmit the generated video files to the collocated data centers and to the non-collocated data centers in cluster 2, the sources in the cluster 2 transmit the generated video files to the collocated data centers and to the non-collocated data centers in cluster 3, and so on, and the sources in the last cluster (cluster 4) transmit the video files to the collocated data centers and to the non-collocated data centers in first cluster (cluster 1). Then, there are 200 within-cluster multicasting groups where each group has 1 source and 2 collocated data centers and 200 cross-cluster multicasting groups where each group has 1 source and 2 non-collocated data centers in total. We will give our simulation setting in Section 6.1.4 and simulation results in Section 6.2.2.

Chapter 6

Simulation and Results

As we mentioned in Chapter 1, we want to explore the possibility of building a distributed multilevel storage architecture (DMSI) by incorporating edge devices and networks for traffic visual surveillance in ITS (See [106]). This is achievable by using network coding in deployment in which there are two main variations, which are deterministic and random paradigms. In our simulation, we want to study the performance of hybrid approach (Section 5.4) for the construction of DMSI, which incorporating both network coding paradigms. The performance of both paradigms can be measured from different tangible metrics such as network throughput, latency and computational cost at network devices. However, there are other subtle aspects which are difficult to be measured, such as the amount of global knowledge required, the complexity of synchronization between network nodes, etc. In this thesis, we focus on measuring the network throughput over the network from sources to sinks for the hybrid approach. In the future extensions, we want to include the study of network latency and computational cost at network devices, so that the trade-offs between various dimensions of both network coding paradigms can be better understood.

We will describe the software architecture of the simulation and its inner components in Section 6.1. Then, we give our simulation results in Section 6.2.

6.1 Simulation Model and Methodology

We will present and elaborate the network model (including network topology, software-defined network, delay analysis, and network link error model), network

coding model (including network coding paradigms and transmission policy) and software architecture for our simulation as follows. We also give our assumptions and analyze the reasons behind those assumptions, and highlight the limitations of the simulation. In the last section, we will describe the simulation of hybrid network coding.

6.1.1 Network Model for Simulation

In this section, we describe the network model being used in our simulation.

6.1.1.1 Network Topology

A common property of many large networks (e.g., the Internet) is that the node connectivities follow a scale-free power-law distribution [101, 102]. This feature was found to be a consequence of two generic mechanisms: first, networks expand continuously by the addition of new nodes, and second, new nodes attach preferentially to sites that are already well connected. Since the IP traffic cameras connect to the network edge and forward the generated video data packet through routers in the network backbone to the data center, we can assume that the network topology of traffic cameras, edge devices, routers and data centers, is also a scale-free network.

In our design, the scale-free network consists of two levels of router nodes, where the top level (AS-level) consists of gateways that interconnect autonomous systems (AS) and the bottom level (RT-level) consists of interconnected routers (RT) within each autonomous system. Moreover, some router nodes from different ASs are interconnecting with each other by using edge connection method such as random or smallest- k degree. We use the hierarchical generation model of BRITE to create the two-level network of routers. The hierarchical generation model first generates the AS-level topology, and then generates the RT-level topologies and connects them to the AS-level. Each level (top or bottom) can use one of the two probabilistic graph models for edge connection, either Waxman’s model [101] or Barabasi-Albert’s model [102], to connect a new node to the set of previously connected nodes. We have simulated several BRITE-generated scale-free networks of 600 (router) nodes.

On top of the BRITE-generated network of routers, the data sources (traffic cameras) are connecting to the leaf nodes (edge devices) or non-leaf nodes (inner routers) in RT-level, and the data sinks (data centers) are connecting to the gateways

of AS-level. For each BRITE-generated network, we connect 200 source nodes to the leaf and non-leaf nodes of the network. Each data source multicasts video data to two data sinks (data centers). Therefore, there is a total of 802 nodes for each network. For simplicity, we assume that all the links have 1% packet lost rate. Moreover, we would like to explore the parameter space by considering the impact of changing the number of source nodes, which is a fixed number in current simulation.

Table 6.1: Parameters for the Scale-free Networks Generated by BRITE

BRITE Network	Edge Connection Method	AS Model	BWdist	N_{AS}	Node Placement	Growth Type	m
1	smallest 3-degree	Barabasi	100Mbps	30	random	incremental	2
2	smallest 3-degree	Waxman(0,15,0.2)	100Mbps	30	random	incremental	2
BRITE Network	Total Router Nodes	RT Model	BWdist	N_{RT}	Node Placement	Growth Type	m
1	600	Waxman(0,15,0.2)	100Mbps	20	heavy-tailed	incremental	2
2	600	Waxman(0,15,0.2)	100Mbps	20	heavy-tailed	incremental	2

Table 6.1 shows the parameters we have chosen to generate the scale-free networks by using BRITE. The BRITE-generated networks are presented in Figure 6.1. We give the meaning of the parameters to generate BRITE network in the following two tables. The detailed description can be found at [96].

Table 6.2: Flat Topology (AS Only or Router Only) Parameters

Parameter	Meaning	Values
N_{AS}	Number of ASs	int $N_{AS} \geq 1$
N_{RT}	Number of nodes in a AS	int $N_{RT} \geq 1$
Model	Network generation model	Waxman or Barabasi-Albert
α	Waxman-specific exponent	$0 < \alpha \leq 1$
β	Waxman-specific exponent	$0 < \alpha \leq 1$
Node Placement	How nodes are placed in the plane	random or heavy-tailed
m	Number of links per new node	int $m \geq 1$
Growth Type	How nodes join the network	Incremental or random
BWdist	Bandwidth assignment to links	float ≥ 0

Table 6.3: Top-down Hierarchical topology Parameters

Parameter	Meaning	Values
Edge Connection Method	Method for interconnecting router topologies	random or smallest- k degree
Total Router Nodes	Number of total router nodes of generated network	$N_{AS} * N_{RT}$

Figure 6.1: The Scale-free Networks Generated by BRITE in Our Simulation.

6.1.1.2 Software-defined Network

SDN designers now face new choices in the design of SDN network; in particular, how centralized or distributed should the network control plane be? Fully physically centralized control is inadequate because it limits (i) responsiveness, (ii) reliability, and (iii) scalability. Thus, designers resort to a physically distributed control plane, on which a logically centralized control plane operates. Large SDN networks will be partitioned in multiple controller domains/subnets; each controller is responsible for one domain/subnet, and the controllers of adjacent domains/subnets may need to communicate to enforce global policies [33]. In doing so, however, they face trade-offs between different consistency models and associated liveness properties [32].

In our model, we assume that there is a logically centralized SDN controller, which has the global knowledge of the network model, while there are many physically distributed SDN controllers, each controls an AS (a domain/subnet) by collecting the information of network states. The communication between the control plane and data plane can be implemented by using TCP protocol to assure the control and feedback delivery. We also assume that the geographical scale of the network in our simulation is at the scale of the city such as Singapore, so that the propagation delay, which is of the order of tens of microseconds, is negligible. The detailed delay analysis will be given in Subsection 6.1.1.3. For the networks of larger geographical scale, we would like to consider in our future works.

6.1.1.3 Delay Analysis

The network latency is composed of four components, which are node processing latency, queuing latency, propagation latency, and transmission latency. The first two components of latency are based on the power of processing node while for the latter two components, propagation latency is based on the scale of the network in terms of travel distance across network links, and the transmission latency is based on the link bandwidths. Queuing latency of a data packet also depends on the network traffic at the time when the data packet is enqueued. In general, the network latency can be summarized in the following equation:

$$l = n + q + p + t, \quad (6.1)$$

where l is network latency, n is node processing latency, q is queuing latency, p is propagation latency, and t is transmission latency.

In our simulation, there are two types of network latencies: first, measurement and monitoring latency between the controller in the control plane and the switches in the data plane, and second, video data transmission latency between the sources (traffic cameras) and sinks (data centers) through the network routers. To use deterministic network coding, the first type of latency is crucial since the SDN controller has to collect the network state information (e.g., bandwidth, delay, etc) across the network in order to compute disjoint edge paths from source to sinks based on the collected network states. The first type of latency may occur in any random network coding scheme which requires global state information to compute multicasting paths [56]. To simplify the simulation setup, we assume that the first type of latency is negligible since we are simulating the video transmission over the network at the geographical scale like Singapore. The second type of latency occurs in both types of network coding schemes, either deterministic network coding or random network coding.

In a network link, the propagation speed is about two-thirds of the speed of light in vacuum [19, 34]. By simple calculation, it is of the order of tens of microseconds for any single data packet transmission at the city scale as Singapore. For example, if a network link has propagation distance of 20km (from either from the west to the central of Singapore or from the center to the east of Singapore), then $20\text{km}/200000\text{kms}^{-1} = 100$ microseconds. Therefore, we can assume that the prop-

agation delay is less than 100 microseconds at all network links in our simulation, which is obviously a very generous assumption. We set the propagation delay as 50 microseconds across all links.

In Singapore, the network bandwidth provided by ISPs to the consumers is about 1Gbps or more for the uplink/downlink speed [100]. Therefore, we are safe to assume that each camera source has at least 100Mbps network bandwidth connecting to the edge devices. We also assume that the network bandwidth across all links of the scale-free network is 100Mbps.

For a video frame of MP4 standard, it is about a few hundred bytes up to a few KB. For a 1-hour video of size 1GB, it is about 17MB per minute (2.267Mbps) in average, so we can expect that the transmission delay is of the order of milliseconds.

6.1.2 Network Coding Model and Delivery Scheme

In this section, we describe our network coding model and delivery scheme (which including transmission policy) over the simulated BRITE network.

6.1.2.1 Video Trace File

We use the video trace file *Verbose_silence.dat* of MPEG-4 provided by the website [93] in our simulation. The video trace files is of 24fps. For a 24-fps video, the time between two frames is about $1/24 \approx 40$ milliseconds. Therefore, we schedule to send a video frame every 40 milliseconds at each data source, where the frame is passed to the network coding module (See Section 6.1.2.3) between the application and transport layer. For random network coding paradigm, we retransmit a transmitted frame at every 2 milliseconds to combat the linear dependency issue incurred by the random coefficients.

6.1.2.2 Transmission Policy

We perform the network coding between the application and transport layer of the UDP/IP model as shown in Figure 6.2. Although other proposals [85, 82, 83] apply network coding between the transport and medium access control layer, we implement it below the application layer to not change other layers within the protocol stack and keep the simulation simple.

Figure 6.2: Implementation of Network Coding Module in UDP/IP model.

6.1.2.3 Network Coding Module

The network coding module (See Figure 6.2) executes two paradigms (either random or deterministic) of network coding.

For random linear network coding, we adopt the framework of [57]. We also compute two edge-disjoint paths from each source to every sink (by using Edmonds-Karp Algorithm) and route the data packets along these paths. We use Kodo [88] to assist our implementation, which is a random linear network coding library under the framework of [57]. Each data source is equipped with one Kodo encoder object, and each router or sink is equipped with multiple Kodo decoder objects to receive and re-encode/decode data packets from the corresponding source.

For deterministic network coding, we adopt the algorithm of [56]. For the first stage of [56], we implement Edmonds-Karp algorithm to find the edge-disjoint paths from each source to every sink. Then we implement the second stage of the algorithm in the network coding layer.

During the encoding at the source, the network coding module treats each video frame as an object to be encoded. We handle the computation of generation size and packet size for random and deterministic network codings differently since it is necessary to tag the random coefficients generated by sources and routers to the transmitted data packets along the paths for random network coding.

For random network coding, we fix the maximum packet size by maximum transferable unit (MTU), which is 1472 bytes, and determine the generation size h_r by

the following formula:

$$h_r = \left\lceil \frac{S_{frame}}{MTU - 20} \right\rceil = \left\lceil \frac{S_{frame}}{1472 - 20} \right\rceil = \left\lceil \frac{S_{frame}}{1452} \right\rceil, \quad (6.2)$$

where S_{frame} denote the size of a video frame and the 20 bytes are reserved for the coefficient tag. Due to the version of kodo library we used, which only supports Galois fields $GF(2)$, $GF(16)$, and $GF(256)$, we limit to the latter two finite fields in our simulation. We don't consider $GF(2)$ since the probability that this finite field incurring linear dependency would be high. If a Galois field $GF(256)$ of size 256 (corresponds to 1 byte) is used, the reserved bytes are able to handle up to a generation size of 20 data packets, which will incur a tag size of 20 bytes; while if a Galois field $GF(16)$ of size 16 (corresponds to half-byte) is used, the reserved bytes are able to handle up to a generation size of 40 data packets, which will also incur a tag size of 20 bytes (See Equation 2.3 and Equation 2.5). Then, each encoded packet size p per generation to be transmitted is determined by the following formula:

$$p = \begin{cases} S_{frame} + S_{tag} & , \text{ if } S_{frame} \leq 1452; \\ (S_{frame} + h_r - r)/h_r + S_{tag} & , \text{ if } S_{frame} > 1452, \end{cases} \quad (6.3)$$

where r is the remainder of generation size h_r divides frame size, $h_r - r$ is the data to be zero padded and S_{tag} is the tag size to be attached to the randomly-encoded data packets. The tag size is computed as follows:

$$S_{tag} = h_r * |GF(\cdot)| \text{ in bytes.} \quad (6.4)$$

Finally, the data packets (of one generation) are randomly-encoded by using Equation 2.2 where the local coefficients are randomly generated at the source or at the routers along the routing paths.

For deterministic network coding, since we don't need to attach the coefficient tag to the data packets, it is unnecessary to reserve a space. Hence, we fix the maximum packet size by maximum transferable unit (MTU), which is 1472 bytes, and determine the generation size h_d by the formula:

$$h_d = \left\lceil \frac{S_{frame}}{MTU} \right\rceil = \left\lceil \frac{S_{frame}}{1472} \right\rceil, \quad (6.5)$$

where S_{frame} denote the size of a video frame. Then, each encoded packet size p per generation to be transmitted is determined by the following formula:

$$p = \begin{cases} S_{frame} & , \text{ if } S_{frame} \leq 1472; \\ (S_{frame} + h_d - r)/h_d & , \text{ if } S_{frame} > 1472, \end{cases} \quad (6.6)$$

where r is the remainder of generation size h_d divides frame size and $h_d - r$ is the data to be zero-padded. Finally, the data packets (of one generation) are encoded by using Equation 2.2 where the local coefficients are generated by Jaggi's construction [56].

At data sinks, the decoding step is the reverse process of encoding by solving a system of equations (See Equation 2.5).

6.1.3 Simulation Architecture

Figure 6.3 shows the high-level abstraction of the software integration of the video trace helper, network coding module (including both network coding paradigms), ns-3 event scheduler and BRITE-generated network in the ns-3 platform.

Figure 6.3: Software Architecture of Integrating Network Coding Module and MPEG-4 Trace File Helper over Simulated BRITE Network by Using ns-3.

The video trace helper is a component that reads each entry of a video trace file of some video in MPEG-4 format (by the *struct* data structure in C/C++) to get the metadata of each video frame such as generated time, frame type and frame size. Then, the helper computes the generation and packet sizes according to Equation 6.2 to Equation 6.6 and passes those parameters to network coding module for further coding process.

After that, the network coding module will linearly encode the video frame according to the chosen paradigm (either random or deterministic), and the event scheduler will schedule a multicasting event and transmit the encoded video frame over the BRITE-generated network to the data sinks according to the generated time of video frame (40 milliseconds per frame). Each multicasting event will ter-

minate as long as all the data sinks receive enough packets to decode the data or the simulation time exceeds the time constraint assigned to each video frame.

6.1.4 Simulation of Hybrid Network Coding

6.1.4.1 Network model

For the hybrid approach (see Section 5.4 for the description), we use the network model in Section 6.1.1 as the basic ingredient to build the interconnection of clusters¹. First, we use BRITE to create 4 clusters where each cluster consists of 150 nodes. We then connect the clusters in such a way that some 4 AS nodes in the cluster i are connected to another 4 AS nodes in the cluster $i + 1$ randomly, and 4 AS nodes in the last cluster are connected to 4 AS nodes in the first cluster randomly. For each cluster, we also connect two data sinks to the AS nodes and 50 sources to some router nodes of the cluster. It follows that there are 200 sources and 8 sinks in total. As a result, the network of interconnected clusters, including sources and sinks, has 808 nodes.

In our simulation, we assume that each video source multicasts a video file to the collocated data centers and the same video file to all the non-collocated data centers of a target cluster. More precisely, we assume that the sources in the cluster i send the generated video files to the collocated data centers and to the non-collocated data centers in cluster $i + 1$ and the sources in the last cluster (cluster 4) send the video files to the collocated data centers and to the non-collocated data centers in first cluster (cluster 1) for the simplicity of the simulation. For example, suppose we have 4 clusters, each cluster has 50 sources, 150 intermediate nodes, and 2 sinks. Then, there are 200 within-cluster multicasting groups (each group has 1 source and 2 collocated data centers) and 200 cross-cluster multicasting groups (each group has 1 source and 2 non-collocated data centers) in total.

In the following table, we give the parameters for the component cluster of 150 nodes. All 4 component clusters have the same parameters.

¹Each cluster is, in fact, a scale-free network which is generated by BRITE based on Barabasi-Albert model or Waxman model.

Table 6.4: Parameters for the Component Cluster of 150 Nodes Generated by BRITE

BRITE Network	Edge Connection Method	AS Model	BWdist	N_{AS}	Node Placement	Growth Type	m
Component Cluster	3-degree	Waxman(0,15,0.2)	400Mbps	6	random	incremental	2
BRITE Network	Total Router Nodes	RT Model	BWdist	N_{RT}	Node Placement	Growth Type	m
Component Cluster	150	Waxman(0,15,0.2)	400Mbps	25	random	incremental	2

We assume that all the links in a cluster have bandwidth 400Mbps and the links connecting clusters have bandwidth 800Mbps. The propagation delay for all the network links is 100 microseconds. For both within-cluster multicasting and cross-clusters multicasting, the retransmission rate is set to 2.0 milliseconds.

To complete the picture, we have also modified the parameter space with the following changes. First of all, we use BRITE to create 4 clusters where each cluster consists of 225 nodes. We then connect the clusters in such a way that some 4 AS nodes in the cluster i are connected to another 4 AS nodes in the cluster $i + 1$ randomly, and 4 AS nodes in the last cluster are connected to 4 AS nodes in the first cluster randomly. For each cluster, we also connect 2 data sinks to the AS nodes and 75 sources to some router nodes of the cluster. It follows that there are 300 sources and 8 sinks in total. As a result, the network of interconnected clusters, including sources and sinks, has 1208 nodes. The following table shows the component cluster of 225 nodes.

Table 6.5: Parameters for the Component Cluster of 225 Nodes Generated by BRITE

BRITE Network	Edge Connection Method	AS Model	BWdist	N_{AS}	Node Placement	Growth Type	m
Component Cluster	3-degree	Waxman(0,15,0.2)	400Mbps	9	random	incremental	2
BRITE Network	Total Router Nodes	RT Model	BWdist	N_{RT}	Node Placement	Growth Type	m
Component Cluster	225	Waxman(0,15,0.2)	400Mbps	25	random	incremental	2

6.1.4.2 Network coding model and delivery scheme

We use the network coding model and delivery scheme described in the previous section (Section 6.1.2) for the hybrid approach as well. As a hybrid approach, we apply deterministic network coding (with $GF(16)$) for data transmissions within a cluster to provide higher throughput and lower computational cost and apply random network coding (with $GF(256)$) for data transmissions across clusters to deal with dynamical network change. Moreover, we assume best-effort delivery and UDP in our delivery scheme (See Figure 6.2 for the illustration).

Given a pair of nodes connected by a channel in the simulated network, we

use the packet retransmission scheme in both network coding paradigms to combat the packet loss over the channel. Intuitively, more packet retransmissions over the channel can combat the packet loss when the probability of packet loss is small. This is because if the first transmission of an encoded packet is lost over the channel, we can just send it again. For an intermediate node in the network, the parent node of the intermediate node will keep sending the cached encoded-packets at a certain rate to the intermediate node as long as the elapsed time lies within the assigned time constraint of a video frame². For the deterministic network coding, the intermediate node will wait until getting enough incoming packets from its parent nodes before linearly combining the received packets and forwarding the encoded packets if the elapsed time lies within the time constraint of a video frame. For the random network coding, the intermediate node linearly combines the cached packets and sends the encoded packets as long as the intermediate node receives a new incoming packet; the sink nodes will decode the video frame provided that the sink nodes receive enough linearly independent packets. Therefore, we don't detect the packet loss at either network layer. Please note that we don't use erasure coding in the current implementation of hybrid approach, however, we intend to include this as our future works.

6.2 Results

In Section 6.2.1, we report the performance of both network coding paradigms, random linear network coding (RLNC) and deterministic linear network coding (DLNC), from the aspect of network throughput - looking at different field sizes ($GF(16)$ and $GF(256)$ for RLNC, and $GF(16)$ for DLNC), as well as different BRITE-generated networks (See Table 6.1). In Section 6.2.2, we give our comparison of hybrid network coding against traditional network coding paradigms over the clustered network (See Table 6.4 and Table 6.5). For that purpose, we compare the performance of these schemes with respect to different network conditions, e.g., network change and churn and packet loss rate. As local user machine, a university PC running Ubuntu 14.04 with 31.3GiB RAM and Intel CPU E5-1650 v4 @

²The assigned time constraint of a video frame should be smaller than the generated time of a video frame. For a 24-fps video, the generated time of a video frame is about 40 milliseconds; for a 48-fps video, the generated time of a video frame is about 20 milliseconds.

3.600GHz $\times$ 12 was used for the experiments. However, we have several new simulations in mind to be done and intend to be pursued as future works, which specifies in Section 7.1.

6.2.1 Comparison between Network Coding Paradigms

First, we compare the performance of deterministic network coding and random network coding employing for the data transmissions in BRITE-generated networks.

6.2.1.1 Network Throughput

Figure 6.4 compares the network throughput of RLNC and DLNC from sources to sinks across different BRITE networks. Given a network topology (e.g., BRITE network sample 1 or BRITE network sample 2), we first fix the network bandwidth (100Mbps) and packet loss rate (1%) across all links. Then we pump the data at a constant bit rate (CBR) from sources to sinks to find the network saturated point. If we cannot get the saturated network by the previous CBR, we increase the CBR; if we get a saturated network, we decrease the CBR to find a non-saturated network. By the previous two steps, we try to search for a near-optimal CBR that the network is saturated. For BRITE network sample 1, we found that the rates are 2.80, 2.69 and 2.40 Mbps for DLNC with $GF(16)$, RLNC with $GF(256)$ and RLNC with $GF(16)$, respectively. Similarly, for BRITE network sample 2, we found that the rates are 3.30, 3.19 and 2.78 Mbps, respectively. From the figure, we can see that DLNC with $GF(16)$ has the highest network throughput, and it is followed by RLNC with $GF(256)$; RLNC with $GF(16)$ has the lowest network throughput among the measured network coding paradigms. This is because using RLNC with $GF(16)$ has a higher probability of causing linear dependency between coding coefficients, and more retransmissions are required to properly decode the data at the sinks and thus more time is taken to download the data at the sinks.

Figure 6.4: Comparison of Network Throughput between DLNC vs RLNC across Different BRITE Networks against Time. Given a network topology, we first fix the network bandwidth (100Mbps) and packet loss rate (1%) across all links. Then we pump the data at a constant bit rate (CBR) from sources to sinks to find the network saturated point.

6.2.1.2 Accumulated Network Traffic between Network Coding Paradigms

(a) BRITE Network 1

(b) BRITE Network 2

Figure 6.5: Comparison of Network Usage between DLNC vs RLNC at All Links of BRITE Network against Time.

Figure 6.5 and Figure 6.6 compares the network usage of RLNC and DLNC from sources to sinks across different BRITE networks. To show the network usage, we send the video trace file *Verbose_silence.dat* [93] from sources to sinks. From Figure 6.5, we can see that RLNC with $GF(16)$ has the highest network usage per second, and it is followed by RLNC with $GF(256)$; DLNC with $GF(16)$ has the lowest network usage among the measured network coding paradigms. Similar to previous figure, RLNC with $GF(16)$ has the highest accumulated network usage, followed by RLNC with $GF(256)$, and DLNC with $GF(16)$ has the lowest accumu-

Figure 6.6: Accumulated Network Usage at All Links of BRITE Network against Time. In this experiment, we send the video trace file *Verbose_silence.dat* [93] from sources to sinks to show the network usage.

lated network usage in Figure 6.6. This is because using RLNC with $GF(16)$ has a higher probability of causing linear dependency between coding coefficients, and more retransmissions are required to properly decode the data at the sink, and thus more data travel across network and increase the network usage.

6.2.1.3 Coefficient Tag Overhead

Since DLNC does not require to attach the generated coefficients to the encoded data packets, as the local coefficients are generated deterministically by using Jaggi's construction, we only report the tag overhead for RLNC. Figure 6.7 compares the tag overhead of RLNC with $GF(16)$ and RLNC with $GF(256)$. We can observe that

Figure 6.7: Total Tags Transmitted across Network.

RLNC with $GF(256)$ has higher tag overhead than $GF(16)$ since the field size of $GF(256)$ is corresponding to one byte while the field size of $GF(16)$ is corresponding to half byte. This verifies the theory that RLNC with smaller field size has lower overhead. However, the linear dependency incurred by RLNC with $GF(16)$ is more severe than RLNC with $GF(256)$, and thus it has higher network usage. Therefore, even RLNC with smaller field size can save the tag overhead, its performance is seriously hampered in network traffic.

6.2.2 Comparison of Hybrid Network Coding against Traditional Network Coding Paradigms

Now, we compare the performance of the hybrid network coding against other network coding schemes over clustered networks. There are three different schemes, which are hybrid, all-random and all-deterministic in comparison. For the hybrid scheme, we apply deterministic network coding with $GF(16)$ for within-cluster multicasting and random network coding with $GF(256)$ for cross-cluster multicasting (see Section 5.4). For the all-random scheme, we apply random network coding, either with $GF(16)$ or with $GF(256)$, for both within-cluster multicasting and cross-cluster multicasting. Similarly, for the all-deterministic scheme, we apply deterministic network coding (with $GF(16)$) for both within-cluster multicasting and cross-cluster multicasting. Note that we use a small finite field (e.g., $GF(16)$) for deterministic network coding, since the number of sinks (2 in each multicasting group) is less than 16, and $GF(16)$ is sufficient for the decoding purpose [56, Theorem 3]. Also, note that we exclude the case of using random network coding for within-cluster multicasting and deterministic network coding for cross-cluster multicasting.

To find the network throughput, we adopt the method described in the previous subsection (see Section 6.2.1). We set 400Mbps as the network bandwidth for all the links inside a cluster and 800Mbps for all the cross-cluster links. Then we pump the data at a constant bit rate (CBR) from sources to sinks to find the network saturated point.

6.2.2.1 Network Throughput under Network Change and Churn

In this subsection, we compare the performance of different schemes under the network change and churn. The rationale that we use deterministic network coding for data transmission within a cluster in our hybrid approach is because of the small network change (say 1% or less) in clusters (local networks) and thus the clusters are stable. Therefore, in our simulation, we evaluate the following three scenarios where we assume that the network change and churn occur on the nodes connecting between clusters:

1. Choose a specific percentage³ of nodes that are down/up per unit time, through-

³Percentage of nodes being a parameter.

out the simulation. We choose 5% in our simulation.

2. There is a static network but at some time point there is a percentage of nodes that go down (a network churn spike). We choose 5% in our simulation.
3. A mix of 1 and 2: the network is static for a period of time, then there is parametrized dynamics in the network for a period of time, and then the network is static again. We choose 5% for both cases 1 and 2.

We first give the results for the network of 808 nodes and then the network of 1208 nodes.

In the scenario 1 (See Figures 6.8), we see that the hybrid scheme performs the best in terms of total throughput⁴ than other network coding schemes under the network change (See Figure 6.8(c)). Furthermore, the hybrid scheme performs about 5% better than the all-random scheme with $GF(256)$ in terms of total throughput. Figure 6.8(a) shows that both the hybrid and all-deterministic schemes have the highest throughput in the case of within-cluster data transmissions. However, the all-deterministic scheme performs badly in the case of cross-cluster data transmissions (See Figure 6.8(b)). This can be attributed to the synchronization of packets at the intermediate routers causing the delay and reduced throughput.

In the scenario 2 (See Figures 6.9), the hybrid scheme remains the best performer in terms of total throughput and it is followed by the all-random scheme with $GF(256)$ and the all-deterministic scheme (See Figure 6.9). The performance of all-deterministic scheme and all-random scheme with $GF(256)$ is close to each other. This is because that the all-deterministic scheme performs better than the all-random scheme in the case of within-cluster data transmissions but the all-random scheme performs better than the all-deterministic scheme in the case of cross-cluster data transmissions so the total is even. All-random scheme with $GF(16)$ has the worst performance in the scenarios 2. This can be explained by the linear dependency nature of all-random scheme since more retransmissions are required to combat the high probability of packet collision for the $GF(16)$ at the sinks.

In scenario 3 (See Figures 6.10), the general trend is similar to the previous two scenarios; the hybrid scheme has the highest throughput among other schemes (See Figure 6.10(c)). All-random scheme with $GF(16)$ remains the worst performer.

⁴Total throughput = within-cluster throughput + cross-cluster throughput.

(a) Within-cluster

(b) Cross-cluster

(c) Total

Figure 6.8: Comparison of Network Throughput of Hybrid Network Coding against All-random and All-deterministic Schemes under Scenario 1 for the Network of 808 Nodes. In the scenario 1, 5% of nodes go down randomly at the start of every second and come up at the end of the second, repeatedly.

(a) Within-cluster

(b) Cross-cluster

(c) Total

Figure 6.9: Comparison of Network Throughput of Hybrid Network Coding against All-random and All-deterministic Schemes under Scenario 2 for the Network of 808 Nodes. In the scenario 2, the network is static before 10 seconds; after 10 seconds, 5% of nodes go down and leave the network at some point of time.

(a) Within-cluster

(b) Cross-cluster

(c) Total

Figure 6.10: Comparison of Network Throughput of Hybrid Network Coding against All-random and All-deterministic Schemes under Scenario 3 for the Network of 808 Nodes. In the scenario 3, the network is static before 10 seconds; after 10 seconds, 5% of nodes go down and leave the network at some point of time, and 5% of nodes go down randomly at the start of every second and come up at the end of the second, repeatedly until 30 seconds.

Now we give the results for the network of 1208 nodes (See Figures 6.11, 6.12, 6.13). We have compared the performance of the hybrid scheme against the other schemes under the three aforementioned network scenarios. In all the three scenarios, the hybrid scheme performs better than the remaining schemes and is about 5% better than the all-random scheme with $GF(256)$. In fact, we expect the total throughput increases with respect to the increased network size. However, our current results do not show this trend. This is because of the sample of network size is too small (e.g., only 808 nodes and 1208 nodes) and cannot be a representative of the realistic scenario showing the increasing trend. All the simulations are run on a single campus PC; we do not use the idea of parallel and distributed simulator across multiple PCs in our simulation [98, 99]. The number of supported nodes is bounded by the available memory of PC. In [99], they used the plain vanilla version of ns-3 and fit 750 thousand nodes in the 32GB RAM of a single PC, which is about 42KB per node. However, in our simulation, the scenario is more complicated; we need to bind the kodo objects and data buffers to each node, which can tremendously reduce the number of supported nodes. In our experience, a network of 808 nodes with a set of kodo objects and data buffers binding to each node can occupy about 1GB to 2GB RAM. In addition, a complete run of the simulation with the network of 808 nodes can take up to 8 hours. Due to the limitation of the simulator to simulate a network with a considerable number of nodes and large packet on a single PC, we are unable to simulate a network of more than 1500 nodes at the moment.

(a) Within-cluster

(b) Cross-cluster

(c) Total

Figure 6.11: Comparison of Network Throughput of Hybrid Network Coding against All-random and All-deterministic Schemes under Scenario 1 for the Network of 1208 Nodes. In the scenario 1, 5% of nodes go down randomly at the start of every second and come up at the end of the second, repeatedly.

(a) Within-cluster

(b) Cross-cluster

(c) Total

Figure 6.12: Comparison of Network Throughput of Hybrid Network Coding against All-random and All-deterministic Schemes under Scenario 2 for the Network of 1208 Nodes. In the scenario 2, the network is static before 10 seconds; after 10 seconds, 5% of nodes go down and leave the network at some point of time.

(a) Within-cluster

(b) Cross-cluster

(c) Total

Figure 6.13: Comparison of Network Throughput of Hybrid Network Coding against All-random and All-deterministic Schemes under Scenario 3 for the Network of 1208 Nodes. In the scenario 3, the network is static before 10 seconds; after 10 seconds, 5% of nodes go down and leave the network at some point of time, and 5% of nodes go down randomly at the start of every second and come up at the end of the second, repeatedly until 30 seconds.

6.2.2.2 Our Proposal under Other Network Setting

To evaluate our proposal under more general network setting, we remove the assumption that we use in Section 6.2.2.1; that is, the network change and churn not only occur on nodes connecting clusters, but also can occur on nodes within these clusters. Similar as Section 6.2.2.1, we evaluate the hybrid approach against other schemes under the following network scenarios with some parameter changes:

1. Choose a specific percentage of nodes that are down/up per unit time (percentage of nodes being a parameter), throughout the simulation. In our simulation we choose 5 nodes per cluster and the total is 20 nodes.
2. There is a static network but at some time point there is a percentage of nodes (a parameter) that go down (a network churn spike). In our simulation we choose 5 nodes per cluster and the total is 20 nodes.
3. A mix of 1 and 2: the network is static for a period of time, then there is parametrized dynamics in the network for a period of time, and then the network is static again. In our simulation we choose 5 nodes per cluster for both cases 1 and 2.

Under the new setting, Figure 6.14 shows that the hybrid scheme is no longer the best performer in terms of total throughput. The reason is the high network change rate in the local networks, which causes the deterministic network coding underperforming in the case of within-cluster data transmissions. Therefore, the total throughput is not so good and close to the total throughput of all-random scheme.

In the scenarios 2 and 3, the total throughput of hybrid scheme performs similarly as scenario 1 (See Figure 6.15 and Figure 6.16). In both the scenarios, the all-random scheme with $GF(256)$ performs better than the hybrid scheme. The performance of hybrid scheme is between the all-random scheme with $GF(256)$ and the all-random scheme with $GF(16)$.

(a) Within-cluster

(b) Cross-cluster

(c) Total

Figure 6.14: Comparison of Network Throughput of Hybrid Network Coding against All-random and All-deterministic Schemes under Scenario 1. In Scenario 1, we assume that there are 5 nodes per cluster (the total is 20 nodes) that are down/up per second.

(a) Within-cluster

(b) Cross-cluster

(c) Total

Figure 6.15: Comparison of Network Throughput of Hybrid Network Coding against All-random and All-deterministic Schemes under Scenario 2. In scenario 2, after 10 seconds, we assume that there are 5 nodes per cluster (the total is 20 nodes) that go down and leave the network.

(a) Within-cluster

(b) Cross-cluster

(c) Total

Figure 6.16: Comparison of Network Throughput of Hybrid Network Coding against All-random and All-deterministic Schemes under Scenario 3. In scenario 3, the network is static before 10 seconds; after 10 seconds, 5 nodes per cluster go down and leave the network at some point of time, and 5 nodes per cluster go down randomly at the start of every second and come up at the end of the second, repeatedly until 30 seconds.

In conclusion, our hybrid scheme works well when the network change and churn happens to the nodes connecting the clusters. Our proposed hybrid scheme performs about 5% than the all-random scheme with $GF(256)$ when the network change happens to the nodes connecting the clusters. This should fit to the realistic scenario where the local network is stable while the connection between different networks are unstable.

6.2.2.3 Network Throughput when No Packet Loss and 0.05 Packet Loss Rate

We compare the network throughput of hybrid network coding against the other two schemes for no packet loss and 0.05 packet loss rate in this subsection. The comparisons are given as Figure 6.17 for no packet loss and as Figure 6.18 for 0.05 packet loss rate.

From Figure 6.17, we see that all-deterministic has better performance in terms of throughput than the other two schemes when there is no packet loss. It followed by the hybrid scheme and all-random scheme has the lowest performance. This is because all-deterministic scheme doesn't require retransmissions when no packet loss but both of the hybrid and all-random schemes requires retransmissions to combat the linear dependency issue.

From Figure 6.18, we see that hybrid scheme has slightly better performance in terms of throughput than the all-random scheme with $GF(256)$ when the packet loss rate is 0.05. It followed by all-deterministic scheme and all-random scheme with $GF(16)$ has the lowest performance. This is because all-random scheme requires more retransmissions to combat linear dependency for within-cluster multicasting when the field size is small, and all-deterministic scheme requires more retransmissions to combat packet loss for cross-cluster multicasting when traveling for long distance and synchronization between upstream and downstream nodes.

Figure 6.17: Comparison of Network Throughput of Hybrid Network Coding against All-random and All-deterministic Schemes when No Packet Loss.

Figure 6.18: Comparison of Network Throughput of Hybrid Network Coding against All-random and All-deterministic Schemes under 0.05 Packet Loss Rate.

6.2.2.4 Network Throughput under Different Packet Loss Rates

In this subsection, we give the performance of the network throughput of these network coding schemes including the hybrid scheme, the all-random scheme with $GF(16)$ or $GF(256)$ and the all-deterministic scheme under different packet loss rates. We study from different angles - looking at the comparison of the average network throughput of these network coding schemes over time against packet loss rates, as well as the evolved network throughput of these network coding schemes against time. The comparison of the average network throughput against packet loss rates is given as Figure 6.19 while the graphs of the evolved network throughput against time are given as Figures 6.20, 6.21 and 6.22.

Figure 6.19: Average Network Throughput over Time against Packet Loss Rates.

From Figure 6.19, we can see that when no packet loss, the all-deterministic scheme performs the best, and it followed by the hybrid scheme and the all-random scheme with $GF(256)$; the all-random scheme with $GF(16)$ performs the worst. This coincides with the conventional wisdom that the deterministic network coding

scheme performs better than the random scheme when the network is stable.

When the packet loss rate is 0.05, the hybrid scheme has the highest average network throughput over time, and it followed by the all-random scheme with $GF(256)$ and the all-deterministic scheme; the all-random scheme with $GF(16)$ has the lowest average network throughput over time. The deterministic network coding for the video transmissions within a cluster is functioning well when the packet loss rate is small, e.g., 0.05 packet loss rate.

When the packet loss rate is 0.10, the all-random scheme with $GF(256)$ performs the best, while the all-deterministic scheme performs the worst. Moreover, the all-random scheme with $GF(256)$ performs better than the hybrid scheme since both the random network coding in the cases of within-cluster multicasting and cross-cluster multicasting of the all-random scheme are more resilient to the packet loss than the deterministic component in the hybrid scheme.

When the packet loss rate is 0.20, the all-random scheme with $GF(256)$ is the only network coding scheme that is still functioning while the other network coding schemes fail to deliver under this packet loss rate. This is because the deterministic network coding in both hybrid and all-deterministic schemes falls apart when the packet loss rate is too high. Moreover, the all-random scheme with $GF(16)$ has serious linear dependency issue, so the sinks cannot get enough linearly dependent packets to decode within the assigned time constraint.

(a) Within-cluster

(b) Cross-cluster

(c) Total

Figure 6.20: Network Throughput against Time for Data transmissions Using Hybrid Approach under Different Packet Loss Rates (no loss, 0.05, 0.10).

From Figure 6.20, we see that the throughput decreases when the packet loss rate increases. When the packet loss rate is more than 0.10, the network is saturated, so there is no result shown.

(a) Within-cluster

(b) Cross-cluster

(c) Total

Figure 6.21: Network Throughput against Time for Data transmissions Using All-random Approach under Different Packet Loss Rates (no loss, 0.05, 0.10).

From Figure 6.21, we see that the throughput decreases when the packet loss rate increases. When the packet loss rate is more than 0.20, the network is saturated, so there is no result shown.

(a) Within-cluster

(b) Cross-cluster

(c) Total

Figure 6.22: Network Throughput against Time for Data transmissions Using All-deterministic Approach under Different Packet Loss Rates (no loss, 0.05, 0.10).

From Figure 6.22, we see that the throughput decreases when the packet loss rate increases. When the packet loss rate is more than 0.10, the network is saturated, so there is no result shown.

6.2.2.5 Further Discussion on Throughput of Hybrid Network Coding w.r.t. Parameter Space

Here, we give a discussion on the throughput of the hybrid scheme against the other schemes⁵ with respect to different parameters over clustered networks. We assume that the network change occur on the nodes connecting between the clusters (See Section 6.2.2.1). The parameters we consider in this discussion are network bandwidth, packet loss rate, network size, network topology/random graph model, etc. For the clustered networks, the total throughput is the sum of the within-cluster throughput and the cross-cluster throughput, so we need to consider the throughput advantage of the component network coding of each scheme in the cases of within-cluster multicasting and cross-cluster multicasting separately.

When the network bandwidth increases, the total throughput of every scheme increases since more data can be sent per unit time. However, we expect that the hybrid scheme performs better than other schemes since the deterministic component of the hybrid scheme for the within-cluster will increase at a steeper rate. Even though the deterministic component of the all-deterministic scheme in the case of within-cluster multicasting will show a similar trend as in the hybrid scheme, however, the deterministic component in the case of cross-cluster multicasting will have bad performance. Therefore, the total throughput will not be better than the hybrid scheme.

When the packet loss rate increases, we expect the throughput of each scheme decreases (see Section 6.2.2.4) since more retransmissions are required to combat packet loss and this increases the operation latency and thus reduces the total throughput. Under this setup⁶, we expect the all-random scheme performs better than the hybrid scheme since both the random network coding in the cases of within-cluster multicasting and cross-cluster multicasting of the all-random scheme are more resilient to the packet loss. However, we predict the hybrid scheme performs better than all-random scheme when the packet loss rate is about 0.05 when extending to the clustered networks according to our results (See Figure 6.18).

⁵All-random scheme and all-deterministic scheme, as mentioned in first paragraph of Section 6.2.2.

⁶We only simulate the packet loss for the network of 600 nodes; we do not simulate the packet loss for the clustered networks.

When the number of nodes increases, resulting in more available links and thus there are more alternative paths from sources to sinks, which implies that more data can be sent from sources to sinks and thus increases the total throughput. Also, we expect that the hybrid scheme will have the best performance in terms of total throughput as the deterministic network coding component in the case of within-cluster multicasting will contribute more throughput than other schemes.

6.2.3 Summary

We summarize our findings in this subsection.

1. By assuming that the network change and churn only happen on nodes connecting clusters, the hybrid scheme performs better than the other network coding schemes in the clustered networks of 808 nodes and 1208 nodes. When we remove the assumption, the hybrid scheme is no longer the best performer. The reason is the high network change rate in the local networks, which causes the deterministic network coding underperforming in the case of within-cluster data transmissions. Therefore, the total throughput is not so good and close to the total throughput of all-random scheme.
2. When the packet loss rate is varied, the best performer of the average network throughput over time of these schemes has changed. For example, the all-deterministic scheme performs the best when no packet loss, the hybrid scheme performs the best when the packet loss rate is 0.05 and the all-random scheme with $GF(256)$ performs the best when the packet loss rate is 0.10. When the packet loss rate is 0.20, the all-random scheme with $GF(256)$ is the only scheme that is still functioning.

Chapter 7

Conclusion and Future Works

We would like to build a distributed multilevel storage infrastructure (DMSI) for the traffic visual surveillance that transfers visual data from traffic cameras to data centers in intelligent transportation system (ITS) with the following design goals: first, support for intelligent surveillance edge devices; second, location-aware resource management; third, reliability and fault tolerance. With that goal, the DMSI uses edge nodes to augment a centralized backbone, facilitated by network coding and emerging fog computing solutions. In detail, the DMSI executes a linear optimization cost model (LO) for the video multicasting from traffic cameras to data centers through the SDN-wired network with the aid of storage helpers where the feasible region of the LO model includes a new set of storage constraints for handling the resource allocation of the storage and network bandwidth between various stakeholders. In our architecture, network coding can be applied to the video flows at the intermediate network nodes along the multicasting paths for better throughput and used to construct storage codes for better storage-bandwidth trade-offs. Moreover, the LO model and the network coding algorithm will be implemented by leveraging OpenFlow, which is a realization of software-defined network (SDN) controller.

Accordingly, we show that for every feasible solution in the feasible region of the LO problem, it is possible to construct a deterministic network code. This network code construction is mainly for the networks at small to medium scales. By using ns-3 and kodo, we have compared the performance of both random and deterministic network codings by looking at the network throughput. The result shows that deterministic network coding performs better than random network coding in terms of network throughput and network usage at the medium-sized networks. We are

aware that the current progression is still far from complete and thus we want to include the following aspects as our future works: upgrading of the current simulation, theoretical analysis, and security of network coding. These aspects will be discussed thoroughly in the next section.

7.1 Future Works

We outline several topics, which we intend to pursue as our future works as follows. The future plan is originally intended for PhD research, however, since I have applied to downgrade to Master of Engineering, the exact scope of the future plan is rescaled to the exploration of hybrid network coding and network simulation.

7.1.1 Network Simulation

We want to extend the current simulation platform (Section 6.1) to study our storage architecture so that we can have a better understanding about the trade-offs between various aspects (e.g., network throughput, network usage, computational cost, network size, link bandwidth, storage, etc) and thus design a hybrid algorithm for networks at different scales based on the simulation results. The simulation extension covers the the following dimensions:

1. Delay analysis and link error analysis. At the current stage, we assume that the propagation delay is of the order of hundreds of microseconds since we are simulating the network coding events in the networks at the geographical scale as Singapore (medium-sized networks). However, one of our goals is to evaluate our system under different network scales. Moreover, we don't install a model for the evaluation of node processing and queuing delays. Those artifacts of the current system may capture an inaccurate description of the network activities, and we want to remove these artifacts in our future works.
2. Storage codes at the network edges. As mentioned in Section 3.2, the possibility to support storage codes (local repairable codes [104] or regenerating codes [103]) is one of the features that our multilevel storage system can provide. Which family of storage codes can provide the best trade-off between storage and bandwidth in our system is the topic we want to consider. Then, the video

data can be cached on the edge storage devices during the traffic peak hours which can reduce the amount of data to be transferred across the networks during the peak periods (See Section 4.2.3). We intend to execute this idea in the coming works.

3. Networks at different scales and networks of different topology. Now, our generated clustered networks only consist of 808 nodes (4 clusters where each cluster has 150 router nodes, 50 source nodes and 2 sink nodes) and 1208 nodes (4 clusters where each cluster has 225 router nodes, 75 source nodes and 2 sink nodes) in the simulation of network change and churn; for the simulation of network with packet loss, the generated network only consists of 802 nodes (600 router nodes, 200 source nodes and 2 sink nodes). We want to explore the parameter space by changing the number of nodes. That is, we can either increase the number of sources, routers, or sinks or increase the geographical scale (aka increase the propagation latency). Moreover, the random graph models used in our simulation are only limited to Waxman model and Barabasi model; therefore, we would like to include other random graph models (e.g., Erdos-Renyi model) in our simulation to generalize our observation. With the different network sizes and different random graph models, we can study the throughput, delay and link error analysis in a more comprehensive way.

7.1.2 Theoretical Analysis

Furthermore, we want to develop the exact optimal solution for the LO problem formulated in Section 5.1 and provide a resource allocation solution for the visual surveillance system over SDN network in IoT in regards to different objective functions. Also, we want to provide a detailed theoretical performance analysis for the separation of storage and transmission flows, as mentioned in Section 5.2, so that we can incorporate local repairable codes [104] or regenerating codes [103] in the design of the surveillance system for the distributed storage layer.

However, the algorithm in Section 5.1 may only be suitable for networks at small to medium scale, due to the limitation of deterministic network coding. Therefore, we would like to develop a hybrid approach for the network at different scales. For the network at small to medium scales, we can use the algorithm for the network

construction in Section 5.2; for the network at medium to large scales, we may use a hybrid approach of random and deterministic network coding paradigms. The threshold to separate which algorithm to use depends on the parameters tested in our simulation. We intend to pursue this as future works.

7.1.3 Security

How to design a secure system for video transmissions over SDN network in ITS relying on network coding method? As pointed out by Prof Alwen and Prof Anwitaman, there are several issues accompany with the existing network-coding-related secure mechanism, which are:

1. The field size using in the current network coding protocols is small (e.g., $\text{GF}(2^4)$ and $\text{GF}(2^8)$ [57, 88, 89]), which causes the network-coding-related secure mechanism insecure;
2. The issue of reused random coefficients in network coding arises concerns, since this will make the encryption scheme easily breakable by solving a system of linear equations.
3. How to make key management and distribution scheme efficient and secure, since there are different encryptions at different cameras, and thus there are different decryptions at different data centers.

Bibliography

- [1] B. Tian, B. T. Morris, M. Tang, Y. Liu, Y. Yao, C. Gou, D. Shen, and S. Tang, “Hierarchical and networked vehicle surveillance in ITS: a survey,” in *IEEE Transactions on Intelligent Transportation Systems*, 2015.
- [2] M. Valera and S. A. Velastin, “Intelligent distributed surveillance systems: a review vision,” in *IEEE Proceedings on Image and Signal Processing*, 2005.
- [3] W. Hu et. al., “A Survey on Visual Surveillance of Object Motion and Behaviors”, *IEEE Trans. Systems, Man, and Cybernetics*, vol. 34, No. 3, 2004.
- [4] R. Nasim, A. Kassler, “Distributed architectures for intelligent transport systems: a survey,” in *Symposium on Network Cloud Computing and Applications*, 2012.
- [5] J. Zhang, F. Wang, K. Wang, W. Lin, X. Xu, C. Chen, “Data-driven intelligent transportation systems: a survey,” in *IEEE Transactions on Intelligent Transportation Systems*, 2011.
- [6] J. W. C. van Lint, “Online learning solutions for freeway travel time prediction,” in *IEEE Transactions on Intelligent Transportation Systems*, 2008.
- [7] J. W. C. van Lint, “A reliable real-time framework for short-term freeway travel time prediction,” in *ASCE Journal of Transportation Engineering*, 2006.
- [8] J. M. Clanton, D. M. Bevly, and A. S. Hodel, “A low-cost solution for an integrated multisensor lane departure warning system,” in *IEEE Transactions on Intelligent Transportation Systems*, 2009.

- [9] D. Schleischer, L. M. Bergasa, M. Ocaa, R. Barea, and M. E. Lpez, “Real-time hierarchical outdoor slam based on stereovision and GPS fusion,” in *IEEE Transactions on Intelligent Transportation Systems*, 2009.
- [10] J.-I. Meguro, T. Murata, J.-I. Takiguchi, Y. Amano, and T. Hashizume, “GPS multipath mitigation for urban area using omnidirectional infrared camera,” in *IEEE Transactions on Intelligent Transportation Systems*, 2009.
- [11] Cisco Global Cloud Index: <http://www.dqindia.com/tag/cisco-global-cloud-index/>.
- [12] S. Paul, J. Pan, and R. Jain, “Architectures for the future networks and the next generation internet: a survey,” in *Comp. Commun.*, 2011.
- [13] C. Labovitz, A. Ahuja, F. Jahanian, “Experimental study of internet stability and wide-area network failures,” in *Proceedings of the International Symposium on Fault-Tolerant Computing*, 1999.
- [14] C. Labovitz, R. Malan, and F. Jahanian, “Origins of Internet routing instability,” in *Proceedings of IEEE INFOCOM*, 1999.
- [15] C. Labovitz, A. Ahuja, A. Bose, F. Jahanian, “Delayed internet routing convergence,” in *IEEE/ACM Transaction on Networking*, 2001.
- [16] A. Elmokashfi, A. Kvalbein, and C. Dovrolis, BGP churn evolution: a perspective from the core, in *Proceedings of IEEE INFOCOM*, 2010.
- [17] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, “Fog computing and its role in the internet of things,” in *Proceedings of Mobile Cloud Computing*, 2012.
- [18] F. Bonomi, R. Milito, P. Natarajan, and J. Zhu, “Fog computing: a platform for internet of things and analytics,” in *Big Data and Internet of Things: A Roadmap for Smart Environments*, Springer, Studies in Computational Intelligence, 2014.
- [19] G. Varghese, “Network Algorithmics,” Elsevier, 2005.
- [20] L. K. Ford, Jr. and D. K. Fulkerson, “Flows in networks,” Princeton University Press, 1962.

- [21] S. Sezer, S. Scott-Hayward, P. Chouhan, B. Fraser, D. Lake, J. Foinnegan, N. Viljoen, M. Miller, and N. Rao, “Are we ready for SDN? Implementation challenges for software-defined networks,” in *IEEE Communications Magazine*, 2013.
- [22] B. Nunes, M. Mendonca, X.-N. Nguyen, K. Obraczka, and T. Turletti, “A survey of software-defined networking: past, present, and future of programmable networks,” in *IEEE Communications Surveys Tutorials*, 2014.
- [23] D. Kreutz, F. Ramos, P. Esteves Verissimo, C. Esteve Rothenberg, S. Azodolmolky, and S. Uhlig, “Software-defined networking: a comprehensive survey,” in *Proceedings of IEEE*, 2015.
- [24] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, “Openflow: enabling innovation in campus networks,” in *SIGCOMM Computer Communication Review*, 2008.
- [25] N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown, and S. Shenker, “NOX: towards an operating system for networks,” in *SIGCOMM Computer Communication Review*, 2008.
- [26] T. Koponen, M. Casado, N. Gude, J. Stribling, L. Poutievski, M. Zhu, R. Ramanathan, Y. Iwata, H. Inoue, T. Hama, and S. Shenker, “Onix: a distributed control platform for large-scale production networks,” in *Operating Systems Design and Implementation, USENIX*, 2010.
- [27] A. Tootoonchian and Y. Ganjali, “HyperFlow: a distributed control plane for OpenFlow,” in *Proceedings of the Internet Network Management Conference on Research on Enterprise Networking, USENIX*, 2010.
- [28] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu, J. Zolla, U. Holzle, S. Stuart, and A. Vahdat, “B4: experience with a globally-deployed software-defined WAN,” in *Proceedings of ACM SIGCOMM*, 2013.
- [29] A. Das, C. Lumezanu, Y. Zhang, V. Singh, G. Jiang, C. Yu, “Transparent and flexible network management for big data processing in the cloud,” in *Proceedings of USENIX Conference Hot Topics Cloud Computing*, 2013.

- [30] A. Arefin, V. K. Singh, G. Jiang, Y. Zhang, and C. Lumezanu, “Diagnosing data center behavior flow by flow,” in Proceedings of IEEE ICDCS, 2013.
- [31] E. Keller, S. Ghorbani, M. Caesar, and J. Rexford, “Live migration of an entire network (and its hosts),” in Proceedings of ACM Workshop Hot Topics Networking, 2012.
- [32] D. Levin, A. Wundsam, B. Heller, N. Handigol, and A. Feldmann, “Logically centralized? state distribution trade-offs in software defined networks,” in Proceedings of ACM SIGCOMM HotSDN, 2012.
- [33] S. Schmid and J. Suomela, “Exploiting locality in distributed SDN control,” in Proceedings of ACM SIGCOMM HotSDN, 2013.
- [34] C. Yu et al., “Software-defined latency monitoring in data center networks,” in Passive and Active Measurement, Springer, 2015.
- [35] J. Yang, E. Yang, Y. Ran, “SDM²Cast: An OpenFlow-based, software-defined scalable multimedia multicast streaming framework,” in IEEE Internet Computing, 2015.
- [36] H. Egilmez, S. Dane, K. Bagci, and A. Tekalp, “OpenQoS: An OpenFlow controller design for multimedia delivery with end-to-end quality of service over software-defined networks,” in APSIPA ASC, 2012.
- [37] Mininet: <http://mininet.org/>
- [38] Floodlight: <http://www.projectfloodlight.org/floodlight/>
- [39] Cisco’s IoT reference model: http://cdn.iotwf.com/resources/71/IoT_Reference_Model_White_Paper_June_4_2014.pdf
- [40] R. Lidl and H. Niederreiter, “Introduction to finite fields and their applications,” Cambridge University Press, 1986.
- [41] F. MacWilliams et al., “The theory of error correcting codes,” New York: North-Holland, 1977.
- [42] I. S. Reed and G. Solomon, “Polynomial Codes over Certain Finite Fields,” in Journal of SIAM 8, 1960.

- [43] M. Sathiamoorthy, M. Asteris, D. Papailiopoulos, A. G. Dimakis, R. Vadali, S. Chen, and D. Borthakur, “XORing elephants: Novel erasure codes for big data,” in *Proceedings of the VLDB Endowment*, 2013.
- [44] H. Schwartz, D. Marpe, T. Wiegand, “Overview of the scalable video coding extension of the H.264/AVC standard,” in *IEEE Transactions on Circuits and Systems for Video Technology*, 2007.
- [45] A. Albanese, J. Blomer, J. Edmonds, M. Luby, and M. Sudan, “Priority encoding transmission,” in *IEEE Transactions on Information Theory*, 1996.
- [46] N. Shacham, “Multipoint communication by hierarchically encoded data,” in *INFOCOM*, 1992.
- [47] M. Effros, “Universal multiresolution source codes,” in *IEEE Transactions on Information Theory*, 2001.
- [48] R. Puri and K. Ramchandran, “Multiple description source coding using forward error correction,” in *Asilomar Conference on Signals, Systems, and Computers*, 1999.
- [49] C. Fragouli and E. Soljanin, “Network coding fundamentals,” *Foundations and Trends in Networking*, 2007.
- [50] R. Bassoli, H. Marques, J. Rodriguez, K.W. Shum and R. Tafazolli, “Network coding theory: A survey,” in *IEEE Communications Surveys and Tutorials*, 2013.
- [51] T. Matsuda, T. Noguchi, and T. Takine, “Survey of network coding and its applications,” in *IEICE Transactions on Communications*, 2011.
- [52] E. Magli, M. Wang, P. Frossard, and A. Markopoulou, “Network coding meets multimedia: a review,” in *IEEE Transactions on Multimedia*, 2013.
- [53] R. Ahlswede, N. Cai, S.-Y. Li, and R. W. Yeung, “Network information flow,” in *IEEE Transactions on Information Theory*, 2000.
- [54] S.-Y. R. Li, R. W. Yeung, and N. Cai, “Linear network coding,” in *IEEE Transactions on Information Theory*, 2003.

- [55] R. Koetter and M. Medard, "An algebraic approach to network coding," in IEEE/ACM Transactions on Networking, 2003.
- [56] S. Jaggi, P. Sanders, P. A. Chou, M. Effros, S. Egner, K. Jain and L. M. Tolhuizen, "Polynomial time algorithms for multicast network code construction," in IEEE Transactions on Information Theory, 2005.
- [57] P. A. Chou, Y. Wu, and K. Jain, "Practical network coding," in Proceedings of the Allerton Conference on Communication Control and Computing, 2003.
- [58] M. Langberg, A. Sprintson, and J. Bruck, "The encoding complexity of network coding," in IEEE International Symposium on Information Theory, 2005.
- [59] M. Wang and B. Li, "How practical is network coding," In IEEE International Workshop on Quality of Service, 2006.
- [60] J. Heide and D. Lucani, "Composite extension finite fields for low overhead network coding: Telescopic codes," in IEEE International Conference on Communications, 2015.
- [61] P. A. Chou and Y. Wu, "Network coding for the internet and wireless networks," in IEEE Signal Processing Magazine, 2007.
- [62] R. W. Yeung, "Multilevel diversity coding with distortion," in IEEE Transactions on Information Theory, 1995.
- [63] D. S. Lun, M. Medard, T. Ho and R. Koetter, "Network coding with a cost criterion," in Proceedings of the International Symposium on Information Theory and Its Applications, 2004.
- [64] D. S. Lun, N. Ratnakar, M. Mdard, R. Koetter, D. R. Karger, T. Ho, E. Ahmed, and F. Zhao, "Minimum-cost multicast over coded packet networks," in IEEE Transactions on Information Theory, 2006.
- [65] D. Traskov, N. Ratnakar, D. S. Lun, R. Koetter, and M. Medard, "Network coding for multiple unicasts: An approach based on linear optimization," in Proceedings of the IEEE International Symposium on Information Theory, 2006.

- [66] A. Eryilmaz and D. S. Lun, “Control for inter-session network coding,” in Proceedings of NetCod, 2007.
- [67] R. W. Yeung and N. Cai, “Secure Network Coding,” in Proceedings of ISIT’02, June 2004.
- [68] A. Le and A. Markopoulou, “Cooperative Defense Against Pollution Attacks in Network Coding Using SpaceMac,” in Technical Report. <http://arxiv.org/abs/1102.3504>
- [69] D. Lee, K. Kim, K. Rhee, “Optimal multicast control for simple network coding,” in IEEE Transactions on Vehicular Technology, 2015.
- [70]] F. P. Kelly, A. Maulloo, and D. Tan, “Rate control for communication networks: shadow prices, proportional fairness, and stability,” in Journal of the Operational Research Society, 1998.
- [71] M. Chen, M. Ponc, S. Sengupta, J. Li, and P. A. Chou, “Utility Maximization in Peer-to-peer Systems,” In ACM Sigmetrics, 2008.
- [72] K. Nguyen, T. Nguyen, and S. Cheung, “Peer-To-Peer Streaming with Hierarchical Network Coding,” in Proceedings of IEEE International Conference on Multimedia and Expo (ICME), 2007.
- [73] D. Vukobratovi and V. Stankovic, “Unequal Error Protection Random Linear Coding for Multimedia Communications,” in Proceedings of IEEE International Workshop on Multimedia Signal Processing (MMSP), 2010.
- [74] M. Halloush and H. Radha, “Network Coding with Multi-Generation Mixing: A Generalized Framework for Practical Network Coding,” in IEEE Transactions on Wireless Communications, 2011.
- [75] J. M. Walsh and S. Weber, “A Concatenated Network Coding Scheme for Multimedia Transmission,” in Proceedings of NetCod, 2008.
- [76] N. Thomos, J. Chakareski, and P. Frossard, “Prioritized Video Streaming Through Randomized Network Coding,” in IEEE Transactions on Multimedia, 2011.

- [77] M. Kim, D. Lucani, X. Shi, F. Zhao, and M. Medard, "Network coding for multi-resolution multicast," in INFOCOM, 2010.
- [78] A. K. Ramasubramanian and J. W. Woods, "Video multicast using network coding," in Proceedings of SPIE VCIP, 2009.
- [79] H. Zhang, M. Chen, A. Parekh, and K. Ramchandran, "A distributed multi-channel demand-adaptive P2P VoD system with optimized caching and neighbor-selection," in SPIE, 2011.
- [80] P. Ostovari, A. Khreishah, and J. Wu, "Multi-Layer video streaming with helper nodes using network coding," in Proceedings of the IEEE International Conference on Mobile Ad Hoc and Sensor Systems, 2013.
- [81] P. Ostovari, J. Wu, A. Khreishah, and N. B. Shroff, "Scalable video streaming with helper nodes using random linear network coding," in IEEE/ACM Transactions on Networking, 2015.
- [82] J. K. Sundararajan, D. Shahand, M. Medard, S. Jakubczak, M. Mitzenmacher, and J. Barros, "Network coding meets TCP: Theory and implementation," in Proceedings of IEEE, 2011.
- [83] M. Kim, T. Klein, E. Soljanin, J. Barros, and M. Medard, "Modeling network coded TCP: Analysis of throughput and energy cost," in Mobile Networks and Applications, 2014.
- [84] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose, "Modeling TCP throughput: A simple model and its empirical validation," in Proceedings of ACM SIGCOMM, 1998.
- [85] S. Chachulski, M. Jennings, S. Katti, and D. Katabi, "Trading structure for randomness in wireless opportunistic routing," in SIGCOMM, 2007.
- [86] S. Liu and B. Hua, "NCoS: A framework for realizing network coding over software-defined network," in Local Computer Networks (LCN), IEEE 39th Conference on. IEEE, 2014.
- [87] D Szab et al., "Network coding as a service," arXiv.org, 2016.

- [88] M. Pedersen, J. Heide, and F. Fitzek, “Kodo: An open and research oriented network coding library,” in Networking 2011 Workshops, 2011.
- [89] N. Marcano, M. Pedersen, P. Vingelmann, J. Heide, D. Lucani, F. Fitzek, “Getting Kodo: Network coding for the ns-3 simulator,” in Proceedings of the Workshop on ns-3, 2016.
- [90] MPEG-4 Part 2: <https://www.iso.org/standard/39259.html>
- [91] MPEG-4 Part 3: <https://www.iso.org/standard/25035.html>
- [92] F.H.P. Fitzek and M. Reisslein, “MPEG-4 and H.263 video traces for network performance evaluation,” in IEEE Network, 2002.
- [93] Video trace file: <http://www-tnk.ee.tu-berlin.de/research/trace/ltvt.html>
- [94] ns3: <https://www.nsnam.org/>
- [95] BRITE: <http://www.cs.bu.edu/brite/>
- [96] <https://www.cs.bu.edu/brite/publications/usermanual.pdf>
- [97] B. Swenson, and G. Riley, “Simulating large topologies in ns-3 using BRITE and CUDA driven global routing,” in International Conference on Simulation Tools and Techniques, 2013.
- [98] K. Renard, C. Peri, and J. Clarke, “A performance and scalability evaluation of the ns-3 distributed scheduler,” in ICST 2012.
- [99] S. Nikolaev, P. D. Barnes, Jr., J. M. Brase, et al. Performance of distributed ns-3 network simulator. in ICST, 2013.
- [100] https://en.wikipedia.org/wiki/Internet_in_Singapore
- [101] B. Waxman, “Routing of multipoint connections,” in IEEE Journal of Selected Areas in Communications, 1988.
- [102] A.L. Barabasi and R. Albert, “Emergence of scaling in random networks,” in Science, 1999.

- [103] A. G. Dimakis, P. B. Godfrey, M. Wainwright and K. Ramchandran, “Network coding for distributed storage systems,” in INFOCOM, 2007.
- [104] F. Oggier and A. Datta, “Self-repairing homomorphic codes for distributed storage systems,” In INFOCOM, 2011.
- [105] J. P. Hespanha, P. Naghshtabrizi, and Y. Xu, “A survey of recent results in networked control systems,” in Proceedings of IEEE, 2007.
- [106] C. Ling, A. Datta, J. Xu, “A Case for Distributed Multilevel Storage Infrastructure for Visual Surveillance in Intelligent Transportation Networks,” in IEEE Internet Computing. To Appear.

Appendix A

My Curriculum Vitae

A.1 Education

Nanyang Technological University, Singapore
Doctor of Philosophy

Aug 2014-Present

School of Computer Science and Engineering

Nanyang Technological University, Singapore
BSc (Hons) in Mathematical Sciences

Aug 2005-Jul 2009

Second Class Upper (G.P.A. 4.0/5.0)

A.2 Working Experience

Western Digital, Singapore

Research Engineer

Aug 2014-Nov 2017

1. Developed a data backup system (MyHybridCloud) relying on a hybrid of public clouds (such as Google Drive, Dropbox) and private clouds (such as Seafiler) for outsourcing data that provides reliability and deduplication supports.
2. Researching on distributed storage system over software-defined network for traffic visual surveillance in intelligent transportation system which incorporates the tools from network coding and optimization.

Nanyang Technological University, Singapore

Project Officer

Nov 2011-Jan 2014

1. Developed a multi-cloud private data backup system (InterCloud RAIDer) for outsourcing data to untrusted cloud storage services which is relying on erasure coding method to provide redundancy, deduplication technique to eliminate identical copies, and proof of data procession to check the data integrity.

A.3 Computer Skills

Application software: Microsoft Office, Latex, BRITE

Proficient Programming language: C/C++, Qt, ns3, Java, Matlab

A.4 Publications

1. C. Ling, A. Datta, “InterCloud RAIDer: A Do-it-yourself Multi-cloud Private Data Backup System,” ICDCN 2014.
2. C. Ling, A. Datta, J. Xu, “A Case for Distributed Multilevel Storage Infrastructure for Visual Surveillance in Intelligent Transportation Networks,” IEEE Internet Computing. To Appear.