A System Architecture to Aggregate Video Surveillance Data in Smart Cities

Wenjie Zhou*, Dola Saha† and Sampath Rangarajan†
*Department of Computer Science and Engineering
Ohio State University, Columbus, Ohio 43210, USA
Email: zhouwe@cse.ohio-state.edu
†NEC Laboratories America, Princeton, NJ 08540, USA
Email: dola@nec-labs.com, sampath@nec-labs.com

***Abstract*—In recent years, we have experienced many initiatives in developing smart cities across the world. In these densely populated cities, video surveillance will play an integral role to ensure the safety of the citizens. The major challenge will be to transport high volumes of data generated from dense deployment of video cameras throughout the city to a central aggregation facility for analysis and storage. To address this problem, we propose a system solution using existing public bus transit system to collect data from the cameras and physically transport it to the bus terminus, to be uploaded to the data center. The system uses heterogeneous wireless network interfaces, where the camera nodes form a mesh network to route the data to the nearest bus stop for offloading. We utilize high capacity links using mmWave devices at bus stops to offload the data to the buses when they make routine stops. We also propose a novel routing protocol, *R2H*, to route the data generated from these cameras to a bus stop while minimizing the time to offload the video data to an incoming bus. The protocol is designed to be aware of the various challenges that are specific to this system like bus schedule and video payload. We have evaluated our system using wireless experiments and simulation using actual bus route maps and times from New York city area. Our analysis shows that *R2H* achieves 60% less end to end delay compared to OLSR under varying video payload and time of the day.**

I. Introduction

In the wake of rapid urbanization [9], it is paramount to ensure the security of the citizens in future smart cities. In such scenarios, video surveillance systems will not only increase the intelligence of the system to proactively reduce the chances of various threats, but also will play an important role in investigation and prosecution after any crime. Studies [18] have shown that video surveillance systems reduce the crime rate in various city areas. Also, big cities have already installed several cameras [5] to increase the surveillance. But these cameras do not have good quality resolution and much video processing is required for any recognition system to work. Nonetheless, there are not enough cameras in the network to provide full video coverage of the different corners of the city. To increase the coverage of the video footage, we envision that future smart cities will be equipped with video surveillance cameras in every intersection of the cities. To cover 360° of vision, multiple high definition cameras has to be installed in every street corners.

Installation of multiple video cameras in every street intersection for better coverage will require installing new wires, which has to be connected to the city's network for the data to be uploaded, and maintenance of the wired network in hostile outdoor environment. Reconfiguration of the cameras and addition of new cameras will pose another dimension of challenge in the system. Also, aggregation of data traffic over a cellular connection is infeasible. Our study shows that in Manhattan area, average number of intersections within a circular region of radius of $0.5Km$ is about 60. A base station serving this area will have to support an aggregate video traffic of upto $4.64Gbps$ for 1080p resolution. Therefore, existing cellular networks will be unable to handle this volume of traffic. To address these issues, we propose a wireless system solution, which utilizes existing public transportation system in the city to transport the video surveillance data to the data centers. Buses stop in a pre-defined location at approximately regular intervals. But they stop for a short duration, during which they will collect the data from the neighboring cameras. So, a high volume of data needs to be transferred within a very short time, which required high communication bandwidth. Also, buses do not stop in every intersection and will not be able to communicate with a camera multiple blocks away. So, we route the data through camera nodes in multiple hops to a nearby bus stop, where the bus can collect the data. This system architecture is described in figure 1.

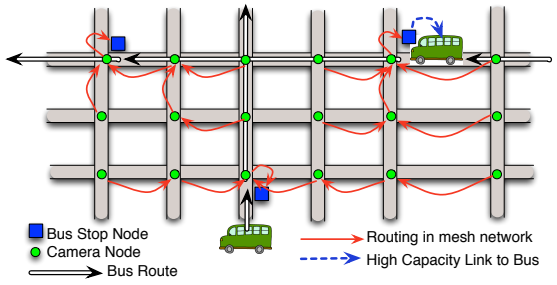


Fig. 1. The system architecture showing streets, where camera nodes form a wireless mesh network to route video data to a nearby bus stop. When a bus visits a stop, data is offloaded from the bus stop node to the bus using wireless high capacity links.

There are three phases in the proposed solution: a) video data is routed to the nearest bus stop, where the camera nodes form a wireless mesh network and not only generate data, but also actively forward data to the other camera nodes until the data reaches a bus stop, b) a bus collects all the data in the stop, when it waits for the passengers using high bandwidth wireless connections, and c) the bus physically carries the data to the terminus where it can upload all the aggregated

data by some high bandwidth connectivity, which could be wired connections. For the first phase of the solution, we propose to use IEEE 802.11n [10] in public safety bands, which can potentially yield upto 600Mbps PHY datarate. IEEE 802.11ac can increase this datarate by multiple folds using MU-MIMO systems. In the second phase, we propose to use 60GHz IEEE 802.11ad [8] based short range links, which can potentially support upto 6.8Gbps physical layer data rate, to collect the data from the bus stops. This heterogeneous network architecture requires a device in the bus stop, which is essentially a collector in the network, having two interfaces, one in public safety band and the other one in 60GHz. In this work, we will focus on the first two phases of the solution.

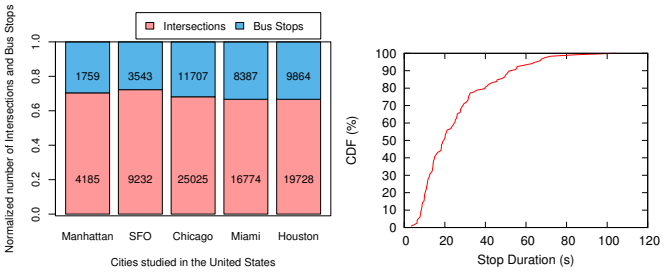
The concept of using public bus transit system for data routing has been proposed in delay tolerant networks [13], [14], [17], [20], [21], [12]. Buses equipped with wireless radios are able to communicate with each other when they are within the radio range. This enables routing data from one corner of a city to another corner [13], [14]. Our proposed solution differs from these schemesi, since in R2H there is no need for routing between buses. Buses that operate between cities and rural areas can transport data from and to remote villages [17], [20], [21], [12]. Although the idea of treating buses as data ferries is similar to R2H, the routing algorithm in these schemes is trivial since the number of buses and stops is limited compared to thousands of them in smart cities. Research has been done in large scale video surveillance on processing local video data to detect object or motion [15], [16], [23]. These studies are orthogonal as well as complementary to R2H, since we focus on collecting the raw video data for passive surveillance and improving intelligence.

The contributions of this work are:

1) A system architecture using heterogeneous network to aggregate high quality video surveillance data from densely deployed cameras in smart cities.
2) A channel condition aware, queue aware and bus schedule aware link state routing protocol to route the data to the bus stops.
3) Measurement of the throughput in outdoor 60GHz links between the bus stop (transmitter) and the bus (receiver) with varying relative position.
4) Data rate estimation in an indoor testbed using IEEE 802.11n wireless USB adapters, which is used as an input to the routing protocol.
5) Performance comparison with OLSR protocol and scalability studies by performing simulations based on real time schedule of buses from the Manhattan area in New York city.

II. Feasibility Study

In this section, we investigate the feasibility of the proposed system solution in various modern cities of the United States. We have studied five cities in the US, as shown in figure 2(a), to find the total number of intersections and bus stops available in each cities. We used OpensStreetMap project's [6] APIs to find the streets and computed the GPS coordinates where they intersect. We used General Transit Feed Specification (GTFS) [1] feeds for each city, which are publicly available, to collect the GPS coordinates of all the bus stops in the



(a) Normalized number of bus stops and intersections in five cities of the United States.

(b) The cumulative distribution function of the duration of a bus waiting in a stop.

Fig. 2. The number of bus stops and intersections, and the waiting duration of a bus at stops.

city Our study shows that only Manhattan area of New York city has more than 4000 intersections. Chicago has more than $25,000$ intersections, and maintaining such a wired network in the often hostile outdoor environments will be challenging. Nonetheless, we notice that for every 3 intersections, there exist a bus stop, which indicates, that the data generated in each intersection may require only 2 to 3 hops to reach a bus stop. As the throughput degrades with number of hops in a mesh network, we argue that fewer hops in the route will not degrade the throughput significantly, which we show with simulation results in section §V .

The proposed architecture depends on two other factors, a) the frequency of the buses, and b) the waiting time of buses in each stop. For the first part, we have collected real time bus schedule for 24 hours from New York City using GTFS-realtime feeds [2] and used it as an input to the simulation, which we discuss in more details in §V. Since these data feeds do not have very fine granularity in time, we have physically boarded the buses in the city and measured the duration a bus waits in a stop with a stopwatch. Figure 2(b) shows the cumulative distribution function (CDF) of the duration, where the median is 20 seconds. We use this value in the simulation, as discussed in §V.

III. System Overview

The proposed system solution will enable the dense unplanned installation of video cameras, which can communicate among themselves in non-line-of-sight scenarios using multihop mesh network. The data can be gathered in a delay tolerant manner as the data moves in multiple hops to the bus, when the bus physically carries the data to the data center. There has been some previous work [13] in routing in delay tolerant network, where the links are not consistent in extreme environments. However, since the cities have high frequency of bus schedule, the data can be delivered with constrained delay guarantees. In the results of the simulation, we have shown that this delay is in the order of a few minutes. In any emergency situation, cellular backhaul can still be used to gather live video streaming data from few of these cameras, whereas the high quality of video data can be collected by the proposed architecture.

We propose *Run-to-Hop (R2H)*, the routing protocol, which is used to route the video data from multiple surveillance cameras to the bus stop. The video data packets aim to *run* or get routed to the bus stop with minimum delay so that they

can *hop* to a bus as fast as possible. The system solution not only aims to minimize the delay in the routing algorithm but also considers the waiting time at each bus stop. The major factors that affect our system are: a) scheduled and real time visits of buses in a stop vary, which changes the queues at each node, b) route closure, when bus stops do not remain as a bus stop, c) changes in channel quality between two nodes, and d) contention among multiple flows that changes depending on the route chosen at source. Our routing algorithm addresses all these factors and models the system to improve the end-to-end delay of the system.

The routing algorithm is based on Link State (LS) routing. The weight of each directed link is calculated using the data rate of the link, the channel contention at the transmitter, and the buffer size at the transmitter. In addition, each bus station has its own weight calculated using the buffer size, the expected time of the next bus, and the expected time the bus arrives at the destination. Based on these weights, each camera node can use Dijkstra's shortest path algorithm to obtain the route.

A. System Model

The network is modeled as a directed graph $G = (V, E)$, where V is the set of nodes in the network including both camera nodes and bus stations, E is a set of directed links connecting nodes in V. Each link $l = (i, j) \in E$ is a directed link from node i to node j. The set $C \subset V$ is the set of camera nodes and the set $B \subset V$ is the set of bus stations in the network. Each node $i \in C$ maintains a link state map M, and a routing table T. Each item in M at node i is denoted as: $(j, q_{ij}, P_{ij}, r_{ij}), (i, j) \in E$, where q_{ij} is the current enqueued data bits to node j at node i, P_{ij} is the expected total number of bits to be delivered to node j in a given time and r_{ij} is the expected UDP data rate from node i to node j. Each item $route$ in the routing table T consists of two parts: expected delivery delay t, and a route to the bus station $R = (C_1, C_2, \cdots, C_K, \cdots, B_j)$, where C_is are camera nodes in the network and B_j is a bus stop. The items in the routing table T are sorted in an increasing order according to the delivery delay t. The first entry in T is used as the route for the currently generated video packets.

B. Routing Algorithm

The objective of the routing algorithm is to route as much data as possible to the buses in a given time. We aim at minimizing the end-to-end delay between the camera node and the bus while avoiding congestion in the network. We use link state routing protocol, where the link weight, w_{jk}, for the link $(j, k), (j \in C, k \in V)$ is defined as:

$$w_{jk} = \left(\sum_{(j,l) \in E} \frac{q_{jl} + P_{jl}}{r_{jl}} \right) + \frac{q_1}{r_{jk}} \tag{1}$$

where q_1 is one segment of data to be transmitted. The first part of the equation essentially captures the delay for already queued packets in node j for all other neighbors l. The second part denotes the delay incurred for one segment of data to be transmitted from node j to node k. This ensures that our routing protocol R2H is aware of the queue conditions in each link of the route.

The link weight for the link $(j, j), (j \in B)$ is calculated using the current queue length (q_j) at the bus station and the expected bus arrival time T_{jk}, where k is the index of the k^{th} expected bus, when all the queued packets at bus stop j can be transmitted to the bus.

$$w_{jj} = T_{jk}, (k-1)Q < (q_j + q_1) < kQ \tag{2}$$

where Q is the expected amount of data that can be delivered from the bus station to the bus. This mechanism ensures that our routing protocol is aware of the queue length at the bus stop and the expected future bus arrival times, which will eventually influence the amount of the data that might be delivered to the buses.

Every node keeps track of the changes in the routing table and broadcasts the first routing entry to its neighbors whenever that entry is changed. Neighboring nodes also update their routing table upon receiving the new route entry. The routing path is determined by the source node and then attached to the packet. All of the nodes on the path follow the route in the packet instead of re-routing packet at every node. In case the path is broken at one relay node, the relay node computes a new route for the packet.

C. Route Update

In this section, we describe the updation process of the routing table, which starts at the bus stop and continues periodically. R2H not only selects the route to a destination, but also chooses the best destination given other conditions affecting the link weight. Hence, unlike other routing algorithms, R2H stores and updates the minimum link weight to the best chosen bus stop and all the packets generated from a node are destined to a bus stop. At any given time, no node in the network is able to route packets destined to an intermediate camera node as the final destination. Instead, the best route to only a bus stop is maintained and all packets generated in that node are routed to that stop. The steps involved in the routing process are as follows:

1) The bus stops update and broadcast their own link weight, $w_{jj}, j \in B$, every T seconds.
2) Each camera node, upon receiving the broadcast message, updates the weight of the route in the routing table destined to the same bus stop according to Equation [1].
3) If this updating changes the first route in the routing table, then the node broadcasts the new route to its neighbors. To avoid any broadcast storm in the network, this broadcast is done only periodically and not triggered by any change in the table.
4) The process continues until no node broadcasts a new route.

The proposed updating process does not need to flood the link weights to every node. Instead, an total link weight metric is maintained for each path. Hence, the updating overhead is decreased compared with other link state routing algorithm such as Fisheye State routing protocol [19]. Figure 3 shows an example of the route update process, where it starts at bus stops B_1 and B_2. Then the information percolates to nodes C_1 and C_3 respectively. Finally, at node C_2, the route is updated based on the minimum weight of the paths and one of the

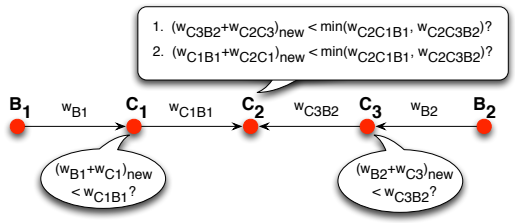


Fig. 3. Route update process, where each camera node broadcasts the link weights periodically and neighbors update the route information.

bus stops B_1 and B_2 is chosen as the final destination. Since every node transmits only one small update packet periodically, the routing overhead in the network is $\mathcal{O}(n)$. In this way, we minimize the overhead of the updating process.

D. Link State Update

To obtain accurate route delay metric, every node $i \in C$ maintains its link state map: $(j, q_{ij}, P_{ij}, r_{ij}), (i,j) \in E$. The current queued number of bits for node j, q_{ij}, can be read from the queue buffer directly. The expected number of bits P_{ij} can be estimated using heuristic method. Learning the expected traffic will not only depend on previous datasets, but also on time of the day, congestion in the network and delays in bus schedule. However, in this work, we choose to use average data generated in the past and skip the learning procedure as our future work.

The technique used to estimate link data rate r_{ij} is non-trivial and critical to link weight estimation. One of the major contributions of this work is to estimate the UDP data rate, r_{ij}, which is affected by both physical layer data rate and the wireless channel contention. The physical layer data rate is again available from the wireless card driver. Since we use commercial wireless card in our experiment, we are not able to obtain the channel contention level directly. Instead we estimate the channel contention in two different cases: 1) channel fully utilized and 2) channel partially utilized.

We first estimate the packet duration t_{ij} for link (i,j) as following:

$$t_{ij} = t_{difs} + t_{avg\_cont} + t_{preamble} + t_{phy\_header} + \frac{b_{ij}}{D_{ij}} + t_{sifs} + t_{ack}, \quad (3)$$

where t_{difs} and t_{sifs} are the DIFS and SIFS durations respectively, which are constants as specified in the standard, $t_{preamble}$, t_{phy_header} and t_{ack} are the durations to transmit the preamble, physical layer header and acknowledgment packets respectively, t_{avg_cont} is the average contention duration, b_{ij} is the number of bits that can be transmitted in each transmission opportunity by node i, and D_{ij} is the physical layer data rate for link (i,j). Let T be the time between two data rate estimation process, t_i be the channel access duration of node i in T and n_i be the number of transmissions by node i in T. Let p_{ij} be the packet reception ratio of link (i,j)

1) Channel Fully Utilized: When the channel is fully utilized and the packet queue of node i is not empty, the total duration of all the other transmissions is $(T - t_i)$. So the contention overhead of each transmission by node i is $t_c = \frac{T-t_i}{n_i}$. Consequently, the UDP data rate for link (i,j) is $r_{ij} = \frac{b_{ij}}{t_{ij}+t_c} \times p_{ij}$.

When the channel is fully utilized and the packet queue of node i is empty, we record the total number of access of the channel in T by each neighboring transmitter. Then the maximum access of node i could be the same as the maximum access of the neighboring nodes, denoted as $N = max\{n_l, (i,l) \in E\}$. So, the UDP data rate for the link (i,j) is $r_{ij} = \frac{b_{ij} \times N}{N \times t_{ij} + T - t} \times p_{ij}$.

2) Channel Partially Utilized: When the channel is not fully utilized, we assume that we can pack the remaining duration of T with transmissions from node i. Assume the channel is busy for a duration of t_{busy} in T. Then the total number of possible transmissions in the link (i,j) is $n_{ij} = \frac{T-(t_{busy}-t_i)}{t_{ij}}$. In this case, the UDP data rate is $r_{ij} = \frac{n_{ij} \times b_{ij}}{T} \times p_{ij}$.

E. Data Rate Estimation

In this section, we describe the data rate estimation algorithm 1, that runs in each node, i, periodically to update $r_{i,j}$, where node j is a neighbor of node i. This is used to calculate the link weight as described in equation 1.

Algorithm 1: Data Rate Estimation Algorithm for Each Node

Input:(i). One hop neighbors information $N_1 = (i, n_{1i}, t_{1i}), 1 \le i \le H_1$. H_1 is the number of neighbors in one hop. n_{1i} is total number of channel access and t_{1i} is channel access duration for neighbor $1i$ in T. (ii). Current PHY data rate r_{phy} for link (i,j) and bits b in each transmission. (iii). The packet reception ratio p_{ij} of link (i,j). (iv). Current node channel information $(n_{cur}, t_{cur}, t_{busy})$.
Output: Estimated UDP data rate r_{udp} for link (i,j).

$t_{ij} \leftarrow t_{difs} + t_{avg\_contention} + t_{preamble} + t_{phy\_header} + \frac{b}{r_{phy}} + t_{sifs} + t_{ack}$
$n \leftarrow max\{n_{1i}\}$
if $t_{busy} \approx T$ **then**
    // channel busy all the time in one hop
    $r_{udp} \leftarrow \frac{b \times n}{n \times t_{ij} + T - t_{cur}} p_{ij}$
    return r_{udp}
if *channel not always busy* **then**
    $n_{ij} = \frac{T-(t_{busy}-t_{cur})}{t_{ij}}$.
    $r_{udp} = \frac{n_{ij} \times b}{T} p_{ij}$
    return r_{udp}

IV. Experimental Evaluation

We divide our evaluation in two phases: a) implement a mesh network to collect video data and route to a bus stop, and b) transmit the data to a bus within a short period of time.

A. Indoor mesh network

1) Network Setup: Since we do not have any licensed spectrum, we deploy the first phase of the mesh network and routing using 5GHz unlicensed frequency. We used the Raspberry Pi [3] as our camera node, equipped with the camera module to capture video and a TP-Link (TL-WDN3200) 802.11n USB

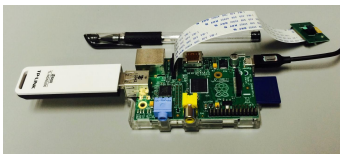
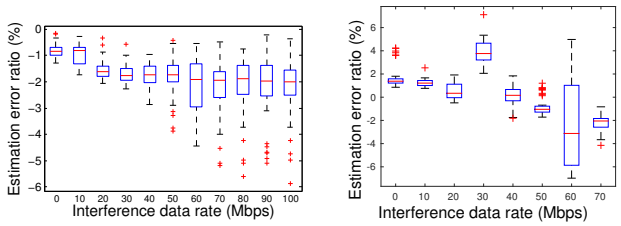


Fig. 4. The camera node, using Raspberry Pi equipped with 1080p video camera and IEEE 802.11n wireless USB adapter.



(a) Estimation error with fully occupied channel.

(b) Estimation error with partially occupied channel.

Fig. 5. UDP data rate estimation in presence of another inerference in testbed.

wireless adapter to transmit the captured video. We setup a small testbed of 10 nodes inside a laboratory environment and was able to receive upto 200Mbps of UDP datarate per link. The adapter uses Ralink RT5572 chips, and we modified the drivers in Linux to obtain all the parameters required for the link estimation, as described in §III-B. However, to show the scalability of the solution, we implement the routing protocol in simulation, which we discuss in section §V.

2) Data Rate Estimation: To verify our link data rate estimation algorithm, we modified the driver for TL-WDN3200 wireless USB adapter to log the required information as described in Algorithm 1. The experiment results are shown in Figure 5. We use four TL-WDN3200 adapters and iperf to create 2 UDP flows. Since the highest physical layer data rate supported by these interfaces is 300 Mbps, the iperf UDP data rate of one link is set to 300 Mbps to make the channel fully utilized. The achievable UDP data rate is around 200 Mbps. Then we increase the data rate of the other link by 10 Mbps every 50 seconds. The difference between our estimated data rate every second and the real achieved data rate reported by iperf is plotted in Figure 5(a). The result shows that the estimation error is around 2% on average. To test the estimation performance under partially utilized channel, we use the average data rate under different interference level in the previous example as baseline comparison. Then we set the iperf data rate of one link to be 30 Mbps constantly and vary the interference level on the other link. The estimation error is shown in Figure 5(b). The result indicates that the error is less that 6%. We believe that our real estimation error should be smaller than this value because our baseline data rate is an average over 50 seconds.

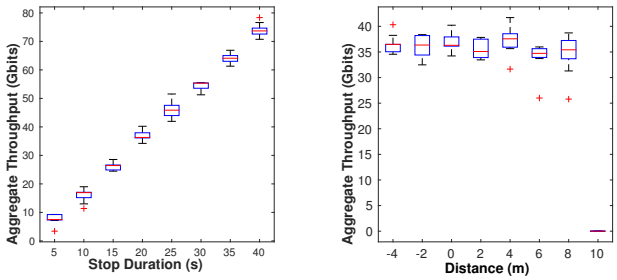
B. Outdoor measurements in 60GHz frequency

In the second phase of the evaluation, we transmit IP packets over commercially available Wireless HD devices, based on our previous work [22]. Although outdoor measurements [11] have been done in 60 GHz realm, this research is the first to deploy short range links over 60 GHz outdoor using beamforming antennas and commercial off-the-shelf (COTS)

(a) Parking lot, where the red rectangle shows the stationary vehicle and the arrows show the path of the moving vehicle.

(b) Top: the stationary and the moving vehicles in the parking lot. Bottom: 60GHz transceiver mounted on a vehicle.

Fig. 6. Outdoor experimental setup.



(a) Data delivered with changing vehicle stop duration

(b) Data delivered with changing distance between the moving vehicle and the stop

Fig. 7. Variation of data delivered by the stationary vehicle to the moving vehicle when it stops.

devices. The intent of this experiment is to understand the amount of the data that can be delivered when the bus stops and the range of communication of the 60GHz links, since these buses may stop farther away from the stop.

We have conducted our experiment in a parking lot with two vehicles. We have mounted a Wireless HD device on top of each vehicle and one of them waits in a parking lot, just like a node in the bus stop. The other vehicle approaches the stationary vehicle, mimicking the motion of a bus, and halts parallel to the stationary vehicle. The Wireless HD device then beamforms and the stationary vehicle transmits data to the other vehicle, until the other vehicle moves away and the link is broken. Figure 6 shows the outdoor experimental setup.

The two factors that determine the amount of data that can be transmitted while the bus comes to a halt are: a) the duration it stops, and b) the distance between the transmitter and receiver devices. The 60 GHz links form directional beams and find the best transmitter-receiver beam pair which will maximize the link quality. We want to see how the delay introduced by this process affects the total amount of data that can be delivered when the bus stops. Also, the orientation of these devices play a major role in deciding whether a link may form or not. In our experiment, we drive one of the vehicles and stop it parallel to the stationary vehicle such that the devices face each other with a gap of about $\approx 2m$ between the devices. Although IEEE 802.11ad [8] is capable of delivering $6.8Gbps$ of physical layer datarate, these COTS

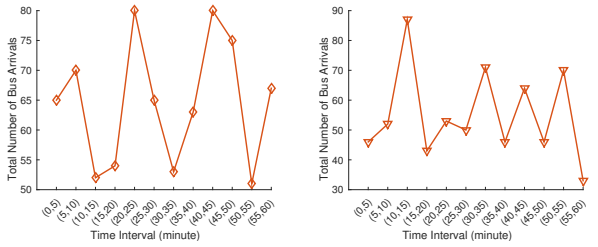
devices do not utilize the full potential of it and we could get upto $2Gbps$ of UDP throughput. Figure 7(a) shows the amount of application data in *Gbits* that is delivered as the duration of the wait time increases from $5seconds$ to $40seconds$. We notice that time required for beamforming is negligible and the data delivered increases linearly with increase in stop duration. We use this information in our simulation to deliver data from bus stop nodes to the buses using real time schedule of the bus. Since the buses in the cities may not stop exactly in the designated location, we also measure the data delivered when the bus stops further away from the stop, keeping the duration constant. From real data collected in the Manhattan area of New York city, we notice the median duration that a bus waits in a stop is $20seconds$, as shown in figure 2(b). So, we keep the duration constant at $20seconds$ for this experiment. Figure 7(b) shows the total amount of data transferred as the distance between the vehicles increase. A negative value in the x-axis indicates that the vehicle moves beyond the designated location and stops after crossing the bus stop. At $0m$, the devices are aligned facing each other. A positive value indicates that the bus comes to a halt before reaching its' designated location. We notice that the data delivered in each of the measured distances are very similar until it reaches $10m$ distance, when the devices cannot form a link. This indicates that the duration to find the best beam for the transmitter receiver pair is not significant and will not play a major role in delivering data to the buses. However, these COTS Wireless-HD devices require to be aligned facing each other for communication as they do not beamform in all directions. Hence, we notice that they cannot form a link at $10m$, as they are not aligned. If they are oriented to face each other, we found the line-of-sight range to be $30m$. However, IEEE 802.11ad devices, will overcome this challenge by sector-level sweep (SLS) and beam refinement protocol (BRP) phases.

V. Trace-driven Simulation

In this section, we evaluate the performance of R2H in ns3 [4] with real data trace. We have modified ns3 to support $400ns$ Guard Interval to achieve upto 72.2Mbps physical layer data rate. With complete implementation of IEEE 802.11n protocol by channel bonding and multiple streams, we can achieve upto 600Mbps of physical layer data rate, which is more than 8 times improvement over our current simulation platform. So, the throughput as shown in our results, will improve significantly if we can leverage the new standards. In this work, our goal is to show the practicality of the system, which can be scaled to support multiple video streams from more number of cameras in the actual deployment.

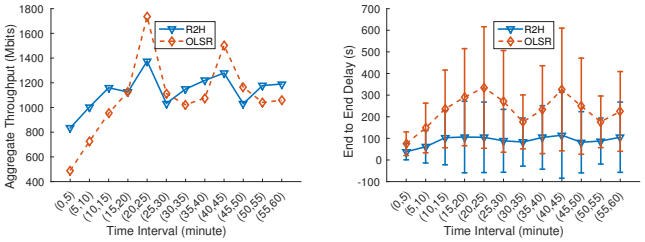
The following datasets are collected from the lower Manhattan area: 1) the GPS locations of all intersections, 2) the GPS locations of all bus stops, 3) the bus schedule of all routes and their expected arrival time at each bus stop, and 5) the duration of a bus stopped at the bus stop and the expected amount of delivered data between the bus and the bus stop as explained in Section §IV-B.

Simulation Setup: We have shown preliminary results in the lower Manhattan area, by defining a rectangular zone with 237 intersections and 50 bus stops. The expected bus schedule and the actual bus arrival time are preloaded to each bus stop. We study the system performance for an hour both



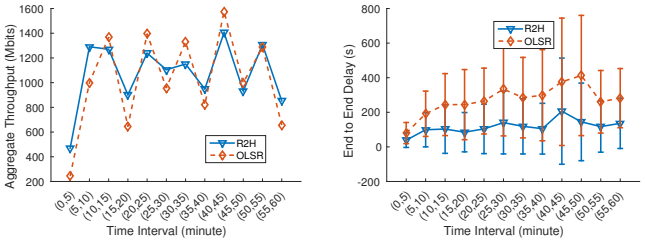
(a) 8 A.M. (b) 10 P.M.

Fig. 8. The number of bus arrivals every 5 minutes starting at 8 A.M. and at 10 P.M..



(a) Aggregate Throughput (b) Delay

Fig. 9. Throughput and Delay of R2H and OLSR with 100Kbps video data generation rate starting at 8 A.M..



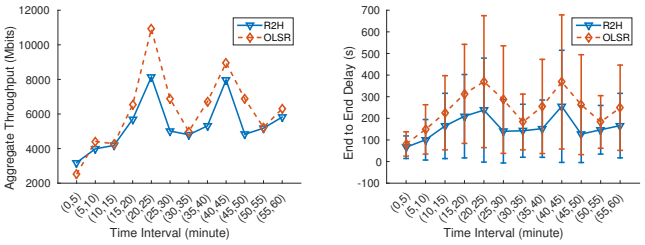
(a) Aggregate Throughput (b) Delay

Fig. 10. Throughput and Delay of R2H and OLSR with 100 Kbps video data generation rate starting at 10P.M..

during the day (8:00 A.M.) and at night (10:00 P.M.). Nodes in the network broadcast their routing and channel access information every 1 second. The data rate generation rate at each intersection is varied between 100 Kbps and 1Mbps. We evaluate aggregate throughput and end to end packet delay for every 5 minutes. Each bus is expected to pick up 35 Gbits of data when visiting a bus stop.

Comparing Scheme: We have chosen the most popular link state based routing protocol, Optimized Link State Routing Protocol (OLSR) [7], to compare with R2H. OLSR is specifically designed for wireless ad hoc networks and is in IETF. It proactively calculates and maintains the shortest hop routes to all nodes in the network based on the flooded topology messages. In our simulation, the data generated by cameras at the intersections are forwarded to the bus stops. So we manually choose the closest bus stop as the destination for each intersection before starting the simulation.

Simulation Results: We study the performance of the proposed protocol by two metrices a) end to end delay and b) throughput, where the time is calculated from the data



(a) Aggregate Throughput (b) Delay

Fig. 11. Throughput and Delay of R2H and OLSR with 1 Mbps video data generation rate starting at 8 A.M..

generated to being delivered to a bus. We first investigate the performance of R2H with 100 Kbps data generated at each intersection starting from 8 A.M.. The number of bus arrivals in every 5 minutes is shown in Figure 8(a), which varies between 51 and 80. Figure 9(a) shows the throughput of R2H and OLSR over a 5 minute window for an hour. The aggregate throughput increases at the beginning because packets accumulate at the bus stops before the first bus arrives. Then the throughput is affected by the number of bus arrivals as indicated by the correlation between the shape of Figure 8(a) and Figure 9(a). Although the average throughput of both scheme are similar, the achieved throughput of R2H is more stable over time and experiences less variation due to the number of bus arrivals. This is because R2H tries to route packets to the bus stops, which expect a bus visiting in the least time. So packets generated in an interval are collected at these bus stops and picked up by the passing buses, which indicates the achieved throughput of R2H mainly depends on the packet generation rate at the camera nodes, which is a constant value. The end-to-end delay of every packet is shown in Figure 9(b). The average delay of R2H is 61% less compared to that of OLSR. Since R2H is aware of bus schedule, it routes packets to the bus stop, where they can be offloaded to a passing bus in the shortest possible time. As for OLSR, it just routes packets to the nearest bus stop, where a bus may arrive after a significant amount of time.

To study the performance of the system under varying inter-arrival times of the buses, the experiment is repeated at 10 P.M.. The number of bus arrivals is shown in Figure 8(b). We observe very similar results as shown in Figure 9. Although R2H only achieves 5% higher throughput than OLSR, it achieves 57.3% less delay compared to OLSR.

Finally we test the performance of R2H and OLSR with higher video data generation rate, at 1 Mbps, starting from 8 A.M.. The delay of R2H is still 35% less than that of OLSR (Figure 11(b)). However, the throughput achieved by R2H is at the most 14% less than OLSR (Figure 11(a)). A close look at the simulation shows that most part of the wireless channel is congested for both schemes with 1 Mbps data generation rate. In presence of congestion in the network, we sacrifice throughput to achieve better end to end delay in the network, by routing the packets to bus stops, where the chance of getting transmitted to a bus is higher.

VI. Conclusion

In this work, we design, implement and evaluate a system solution to aggregate video surveillance data in future smart cities using heterogeneous wireless networks. The system enables unplanned dense deployment of cameras to ensure safety in the smart cities. It is designed around existing public transportation systems to deliver video data for aggregation with a maximum median delay of 5 minutes. The proposed routing protocol can be extended to route the video data to multiple aggregation points with internet connectivity, like bus stops, which further reduces this time for immediate analysis. We also show the robustness of the system against variable inter-arrival time and network congestion. In future, we would like to estimate the bounds for capacity and delay of the system in a dense deployment across various cities in the world.

References

[1] The general transit feed specification (gtfs), https://developers.google.com/transit/.

[2] http://datamine.mta.info/.

[3] http://www.raspberrypi.org/.

[4] The ns-3 network simulator. [online]. available: http://www.nsnam.org/.

[5] NYC surveillance camera project, http://www.mediaeater.com/cameras/.

[6] Openstreetmap project, http://www.openstreetmap.org/.

[7] Optimized link state routing protocol (olsr), 2003.

[8] Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 3: Enhancements for Very High Throughput in the 60 GHz Band , 2012.

[9] World population prospects, the 2012 revision, http://esa.un.org/wpp/, 2012.

[10] Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, March 2012.

[11] S. Deng, C. J. Slezak, G. R. MacCartney Jr, and T. S. Rappaport. Small Wavelengths–Big Potential: Millimeter Wave Propagation Measurements for 5G. *Microwave Journal*, 57, 2014.

[12] K. Fall. A Delay-tolerant Network Architecture for Challenged Internets. In *Proc. of ACM SIGCOMM*, 2003.

[13] S. Jain, K. Fall, and R. Patra. Routing in a Delay Tolerant Network. In *Proc. of ACM SIGCOMM*, 2004.

[14] J. G. Jetcheva, Y. chun Hu, S. Palchaudhuri, A. Kumar, S. David, and B. Johnson. Design and Evaluation of a Metropolitan Area Multitier Wireless Ad Hoc Network Architecture. In *WMCSA*, pages 32–43, 2003.

[15] C. L., B. C., A. J.M., C. B., and S.-E. A. A Semantic Autonomous Video Surveillance System for Dense Camera Networks in Smart Cities. 2012.

[16] H. Liu, S. Chen, and N. Kubota. Intelligent Video Systems and Analytics: A Survey. *Industrial Informatics, IEEE Transactions on*, Aug 2013.

[17] S. Naidu, S. Chintada, M. Sen, and S. Raghavan. Challenges in Deploying a Delay Tolerant Network. In *Proc. of ACM CHANTS*, 2008.

[18] J. A. M. Nancy G. La Vigne, Samantha S. Lowry and A. M. Dwyer. Evaluating the Use of Public Surveillance Cameras for Crime Control and Prevention, 2011.

[19] G. Pei, M. Gerla, and T.-W. Chen. Fisheye State Routing in Mobile Ad Hoc Networks. In *In ICDCS Workshop on Wireless Networks and Mobile Computing*, pages 71–78, 2000.

[20] A. Pentland, R. Fletcher, and A. Hasson. DakNet: Rethinking Connectivity in Developing Nations. *Computer*, 37(1), Jan 2004.

[21] A. Seth, D. Kroeker, M. Zaharia, S. Guo, and S. Keshav. Low-cost Communication for Rural Internet Kiosks using Mechanical Backhaul. In *Proc. of ACM MobiCom*, 2006.

[22] X. Tie, K. Ramachandran, and R. Mahindra. On 60 GHz Wireless Link Performance in Indoor Environments. In *Proc. of PAM*, 2012.

[23] M. Zabocki, K. Gociewska, D. Frejlichowski, and R. Hofman. Intelligent Video Surveillance Systems for Public Spaces A Survey. *Journal of Theoretical and Applied Computer Science*, 2014.