Spider: Next generation Live Video Analytics over Millimeter-Wave Networks

Zhuqi Li [1,2], Yuanchao Shu [1], Ganesh Ananthanarayanan [1], Longfei Shangguan [1], Kyle Jamieson [2], Victor Bahl [1]

[1] *Microsoft Research* [2] *Princeton University*

Abstract

Massive video camera networks are now driving innovation in smart retail stores, road traffic monitoring, and security applications, and realizing live video analytics over these networks is an important challenge that Wi-Fi and cellular networks alone cannot solve. Spider is the first live video analytics network system to use a multi-hop, millimeter-wave (mmWave) wireless relay network to realize such a design. To mitigate mmWave link blockage, Spider integrates a separate low-latency Wi-Fi control plane with the mmWave relay data plane, allowing agile re-routing without suffering the penalty of mmWave beam searching for the new route. With the objective of maximizing video analytics accuracy (rather than simply maximizing data throughput), Spider proposes a novel, scalable flow planning algorithm that operates over hundreds of cameras to simultaneously calculate network routes, load-balance traffic, and allocate video bit rates to each camera. We implement Spider in a mmWave camera network testbed, comparing its object, text, and face detection recall performance against the state-of-the-art wireless video analytics system. Under different video analytic tasks, Spider improves recall by 41.5%—52.9% on average. Further experiments demonstrate Spider's high scalability and gradual degradation under node and link failures, with a 57% reduction in average failure recovery time.

1 Introduction

Live video analytics on camera streams are central to important applications like smart retail stores, road efficiency and safety, and security monitoring [13–15, 32]. Video analytics deployments stream videos to on-premise edge servers [1, 2], which are equipped with GPUs for executing the vision models involved in their processing. The need for high video analytics accuracy drives a push for higher resolution and video frame rates: prior work shows up to 2.2× improvements in accuracy for object detection with 4K video resolution and frame rates of 60 frames/s [36].

On the one hand, higher resolutions and frame rates drastically increase bit-rate requirements of video streams. On the other hand, Wi-Fi links do not have enough capacity to stream multiple high bit-rate videos (to the edge server for analytics). Indeed transmitting bulky video streams over Wi-Fi links would crowd the over-utilized Wi-Fi band and cause severe interference and delay to other Wi-Fi users. Connecting the cameras with wired networks is cumbersome at scale—a typical enterprise building has 100 to 200 cameras [14, 19]—while also would reduce deployment flexibility. Millimeter-wave (mmWave) networks, with capacities of multi-Gbits/second (*e.g.,* 802.11ad WiGig at 60 GHz), are a promising option for streaming high bit-rate videos for video analytics. However, they have well-documented limitations: (*i*) their high throughputs depend on clear line-of-sight, thus making them susceptible to moving objects in the building, and (*ii*) their throughputs drop drastically beyond *ca.* 10 m, considerably shorter than typical distances between cameras and edge servers in deployments.

Spider: A mmWave video analytics network design. We propose *Spider*, a live video analytic network design featuring cameras fitted with WiGig radios and constructing a *multi-hop relay* network of the cameras, as shown in Figure 1. Spider fits WiGig radios to edge servers while cameras *relay* their video streams to one of the edge servers for analytic processing. Cameras that are close to the edge servers communicate directly with the edge servers. Farther-away cameras pass their streams to other *relay* cameras that in turn forward them to the edge server. As a result, network links of the relay cameras closer to the edge servers will experience higher utilization and congestion.

While there have been extensive studies on building single-hop mmWave networks [22, 23, 30, 43, 50], less effort has been focused on bulky video streaming over multi-hop mmWave networks. Constructing a mmWave relay network is difficult because if one link in such a network becomes disconnected as a result of physical blockage [16, 52], the narrow beam width of that link's endpoint radios means that the endpoints would then have to search for a possible alternate reflection path or alternate route altogether in order to reconnect the network, This is problematic because the node being connected to may be busy itself with ongoing data transmission, and thus may not know that it is needed to participate in the search. Such a search, during which the network remains disconnected, also inevitably involves throughput, latency, and jitter penalties [23, 40, 52]. Furthermore, the required step of broadcasting routing information to each node's neighbors in the mmWave relay network is challenging, since each node's radio has to beam towards each of its neighbors in order to exchange routing information, which is costly in time, as mentioned. The conclusion is that we require a separate *control plane* that addresses these

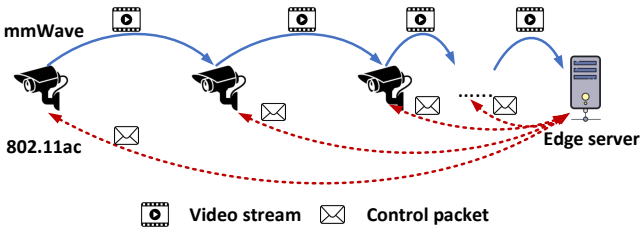


Figure 1: Spider's data flow and control flow. High volume video streams are transmitted over mmWave links (solid line). Short control packets are forwarded over 802.11ac Wi-Fi links (dash line).

issues more effectively than mmWave links.

For the purpose of building the proposed Spider relay network, in which multiple cameras offer video stream load into the network, multiple streams need to share a single mmWave link. This in turn means that there must be some mechanism to allocate the capacity of shared link among individual video streams. In other works, we require a scalable *flow planning* algorithm that operates over hundreds of cameras and thousands of multi-hop links among them. For the same reasons mentioned above, Spider needs to leverage the separate control plane to disseminate instructions to all the cameras in the network, so that they can implement the routing topology and video bit-rate allocations that our flow planing algorithm generates. A further challenge is the dynamic nature of the mmWave wireless channel, which pushes us to develop robust bandwidth estimation mechanism to avoid disrupting live video flows. We also need to dynamically calculate the relationship between the bit-rate of the videos and the accuracy of the video analytics outputs. To address these challenges, Spider introduces the following three design techniques:

1) Network flow planner: Flow planning on multi-hop relay networks is a well-studied problem [18, 20, 21, 29, 37]. However, our objective and design choices differ in two crucial aspects. First, unlike prior work, we do *not* optimize for the network throughput of the flows. Instead, our objective is to maximize the *video analytics accuracy*. As a result, our solution often selects routes and link allocations that are sub-optimal for network throughput but lead to better video analytics accuracy (§4.1). Second, we design a centralized heuristic that uses all the link bandwidths between cameras and globally optimizes for our objective. This is in contrast to the decentralized designs of prior solutions where nodes make their own neighbor discovery and routing choices.

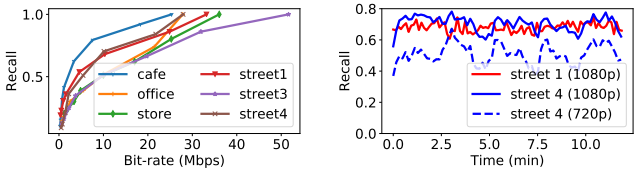
Our heuristic optimizes both network-layer video stream routing and application-layer video bit-rate allocation. Specifically, it first builds an application-independent routing tree based on the intuition that load balancing among relay nodes yields a higher overall video analytics accuracy. It then leverages the unique bandwidth-accuracy tradeoff in video analytics (that varies across cameras and with time), and builds a mixed integer linear programming (MILP) model to decide the application-dependent bit-rate configuration to maximize the overall video analytics accuracy.

2) Separate control and data planes: For the aforementioned reasons, Spider cameras report link bandwidths to the controller, while the controller pushes down the routes for the cameras to use along with the video bit-rates to offer. To communicate these control messages, Spider uses 802.11ac Wi-Fi to achieve more reliable dissemination of control information. We design the control protocols such that their capacity overheads are limited, and thus they do not significantly disrupt existing Wi-Fi traffic.

3) System Design: We have designed a multi-hop relay system of WiGig cameras, Spider. Spider's design does not involve continuous and expensive probing traffic for bandwidth estimation. Given the directional nature of WiGig links and expensive beam searching to initiate new links, any continuous probing would severely disrupt the flows of live videos. Instead, Spider adopts a *reactive* approach to bandwidth estimation by detecting when existing links have reduced throughputs or blockage, and only then probing for new links. Spider also periodically profiles to obtain the relationship between the bit-rate of the video and its impact on the accuracy of the outputs. In doing so, it trades off the timeliness of the video profile for the bandwidth cost for sending video segments of the highest frame-rate and resolution.

We implement Spider in a university building testbed, with 11 distributed camera nodes and stream 55 4K video streams concurrently to the edge server. In further investigate the performance of Spider in a larger environment, we conduct trace-driven simulations in an enterprise environment with 330 camera nodes. Spider achieves improvement on video analytic accuracy by 41.5%—52.9% on testbed and by 26.5%–39.3% with trace driven simulation with enhanced robustness and minimal interference to the existing wireless users.

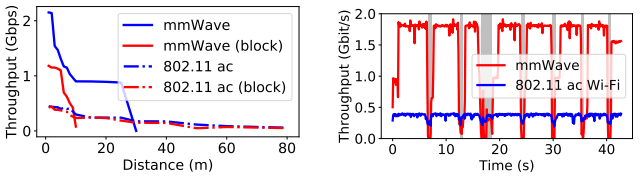
Contributions: We make the following contributions:
1) We identify the tension between large camera networks and wireless link throughput, and propose a relay architecture of WiGig cameras for next-generation video analytics.
2) We devise flow planning algorithms that maximize video analytics accuracy at runtime with joint consideration of network fluctuations and video content variations.
3) We build an 11 node, real-world camera network and evaluate multiple live video analytics applications end-to-end using both testbed and large-scale trace-driven experiments.



(a) Detection recall in different scenes.

(b) Impact of video content variation across time and cameras.

Figure 2: Motivating experiments for video benchmarking.



(a) TCP throughput at different link distances.

(b) TCP throughput variation due to human blockage.

Figure 3: Impact of distance and blockage on 802.11ac Wi-Fi and 802.11ad mmWave links.

2 Background and Motivation

A network featuring live video streaming over wireless networks would substantially lower system cost and increase deployment flexibility, but at the same time, such a design poses challenges in network routing, due to the unique characteristics of video content and wireless links. In this section, we explore the design space of mmWave relay networks in the context of video analytics.

2.1 Video analytics background

While state-of-the-art machine learning models have shown great success in video analysis applications, their performance is highly sensitive to variations in video content and quality [47]. We conduct benchmarks to understand the impact of video content variation across time and cameras.

Bandwidth-accuracy relationship. In this experiment, we collect six videos from the Internet (each lasting for 12 minutes) and down sample them into different resolutions. We then run a state-of-the-art object detection algorithm (YOLOv3 [35]) on these videos to detect objects of interest. These video clips are all from potential Spider application scenarios (*e.g.,* a cashierless store, traffic monitoring, and video surveillance), covering both indoor (cafe, office, and retail store) and outdoor (four different street scenes) settings.

Figure 2(a) shows the object detection recall of the YOLOv3 model under different video bit-rates, translated from various video resolutions. Overall detection recall grows with increasing video bit-rate, indicating that allocating more bandwidth (Spider's network resource) would yield a higher video analytic recall. The marginal gain of detection recall, however, decreases gradually with increasing video bit-rate. On the other hand, detection recall varies significantly across different video clips (scenes) in the same video bit rate settings. For instance, although recall in both office and cafe reaches 98% at 25 Mbps, low bit-rate is more amenable to object detection in cafe—at 6 Mbps, the recall gap grows to 38% (80% *vs.* 42%). Similar results (on both recall and precision) are also reported in other work on video analytics [26, 49]. These results motivate us to prudently tune knobs (*e.g.,* resolution, sampling rate) in video analytics and allocate network bandwidth in order to maximize the overall video analytics performance across different cameras.

Video content variation across time and cameras. In this experiment, we run YOLOv3 on two 12-min video clips and report real-time object detection recall every ten seconds. The result is shown in Figure 2(b). As can be seen, object detection recall fluctuates with time and differs between cameras. This is not surprising as the video content (*e.g.,* the number of people waiting in line to get a coffee) could change drastically in minutes. Nonetheless, detection recall on videos from the same camera but with different resolutions (1080p *vs.* 720p) shows a strong correlation over time. Temporal variations of video analytics performance necessitates a dynamic resource allocation and scheduling design that keeps up with the change of video content.

2.2 Wireless video analytics needs mmWave links

Video streams demand more link throughput than normal audio or text traffic, and so we experiment to understand the characteristics of two candidate wireless technologies for Spider's video analytics, 802.11ac Wi-Fi and 802.11ad WiGig (mmWave).[1] We first measure the throughput of these two technologies in both line-of-sight (LOS) and non-line-of-sight (NLOS) environments (Figure 3(a)). Here we place a Dell E7440 laptop (as the transmitter) and a Netgear Nighthawk X10 router (as the receiver) in an office building, and measure TCP throughput at different link distances. We conduct the experiment at midnight to minimize interference from nearby Wi-Fi. In agreement with previous studies [16] our mmWave links achieve an order of magnitude higher throughput than Wi-Fi links in line-of-sight conditions, but drop much faster over distance: the throughput drops below Wi-Fi throughput when the link distance is over 28 m. In non-line-of-sight condition (dry wall blockage), as we expected, mmWave link experiences severe throughput decline due to strong signal reflections—link throughput drops drastically to zero at around 10 m. In contrast, the blockage has only mild impact on 802.11ac Wi-Fi due to its longer wavelength.

Enterprise deployments of wireless video networks are

[1]We exclude 802.11ax from the comparison, since it has similar throughput and takes up same frequency band as 802.11ac.

vulnerable to blockage. For instance, cameras in cashierless stores are mounted on the shelves; hence wireless links there are prone to be blocked by customers that move inside the store. To understand the impact of human blockage on wireless links, we invite a volunteer to block the LOS path of a three-meter wireless link occasionally and measure the impact on link throughput. We observe that the link throughput of mmWave drops aggressively to almost zero as the volunteer passes through and blocks the wireless link (grey area in Figure 3(b)). In contrast, there is less throughput variation (tens of Mbps drop only) on 802.11ac link when the volunteer blocks the LOS link.

Considering the application scenarios where hundreds or even thousands of cameras are densely deployed (*e.g.,* cashierless store), connecting these camera nodes with low-throughput 802.11ac Wi-Fi links would inherently limit the bit-rate of each video stream, yielding a low video analytics accuracy (§2.1). The high volume video streams produced by these cameras would also saturate the narrow Wi-Fi band, which inevitably introduces significant interference to legacy Wi-Fi users as well as other camera nodes (as experimentally demonstrated in Figure 14 in §6).

Towards a multi-hop relay of WiGig cameras. The foregoing measurements in this section suggest that we ought to adopt mmWave links for wireless video transmission. However, these links fall short of requirements both with regards to transmission distance and with regards to reliability. To cope with these challenges, we further require a multi-hop mmWave *relay* network where each camera node serves as not only a video producer, but also as a relay to forward other cameras' video streams.

3 Spider Architecture

This section provides an overview of Spider. We first present the system workflow (§3.1), followed by the design of the data and control planes, as well as the network's control functionality (§3.2). We then describe video bit rate accuracy profiling and link throughput estimation functionality (§3.3).

3.1 Workflow of cameras and edge servers

Spider consists of a number of WiGig camera nodes and one or more edge nodes on-premises. Edge nodes are equipped with GPUs which execute DNN inference for video processing. Cameras connect to edges via wireless links, while edge nodes are connected through the wired network. When multiple edge servers are present in the network, Spider uses Raft [33] to elect a master server, which runs the scheduler and dispatches video processing workload among all edge servers dynamically (§4.5).

Camera node. Each camera node is equipped with two WiGig radios, one for sending and another for receiving. Camera nodes produce live videos and work hand-in-hand to forward other cameras' video streams to an edge server through the *data plane*. In addition, each camera also produces high-resolution video samples periodically to send to the master edge node for resource-accuracy profiling. To facilitate network management, camera nodes estimate link throughput of all connected links, and periodically report it to the edge server through the *control plane* (Figure 1).

Edge server. Each edge server forwards live video streams directly to its *analytics engine* for video content analysis. To deal with video content variations (§2.1) and link throughput dynamics (§2.2), the master edge server periodically profiles high bit-rate video samples from each camera and updates its resource-accuracy profile. Resources that Spider trades off include both compute budget (across all edge servers) and network bandwidth budget. Based on the latest resource-accuracy profile, the master edge server runs a reactive *scheduler* to reschedule network flows (§4.3) and reallocate bandwidth to each camera (§4.4), based on the camera configurations (*e.g.,* resolution, frame rate) chosen.

Packet flow. There are two types of packets in Spider: data packets and control packets. Data packets transport live video streams from cameras to an edge server. Control packets convey information for system bootstrapping, flow planning, and network status updating, and comprise:

(1) **Peer-discovery packets** discover camera nodes in bootstrapping stage.
(2) **Bitrate-control packets** re-configure video bit-rate on each camera through control plane.
(3) **Route-control packets** reschedule forwarding path for video streams.
(4) **Network-status packets** report link throughput and live video segments to edge servers.

3.2 Decoupling the control and data planes

Since mmWave links are susceptible to attenuation and blockage [16, 52], transmitting control messages over mmWave relay links could lead to a high latency and packet loss ratio. In Spider we therefore design a hybrid network to decouple the control plane from the data plane. This network comprises an 802.11ac Wi-Fi control plane (solid blue lines) and an 802.11ad mmWave data plane (dashed red lines) working on two different frequency bands, as shown in Figure 1.

Spider adopts a centralized control mechanism to manage the video network, allowing it to better plan its data flow, control the network topology, and optimize video analytics performance over the entire network. Centralized control enables the edge server to guide each camera node to switch its relay without exhaustive beam searching, which could take up to 1.5 seconds in 802.11ad [23, 40, 52]. Spider produces on average only five Kbit/s control traffic over the

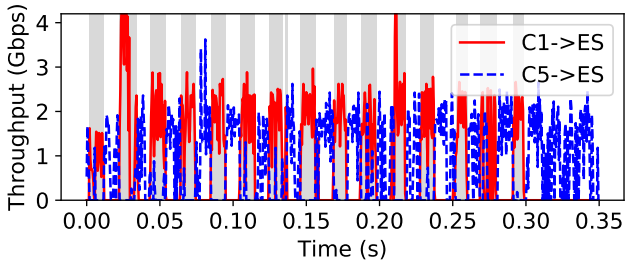


Figure 4: Two camera nodes $C1$ and $C2$ take turns to transmit video streams to the edge server (ES).

entire relay network. Such a small amount of Wi-Fi traffic has negligible effect on legacy users in the same Wi-Fi band, as we experimentally demonstrate in §6.5.

3.3 Video profiling and link throughput estimation

To cope with variation of video analytics accuracy across cameras and time, the master edge node periodically runs video processing algorithms on high bit-rate video samples from each camera and updates a resource-accuracy profile. The edge server makes a careful trade-off between the bandwidth cost of sample video segments and the freshness of profile. We quantify the impact of profiling interval in §6.5.

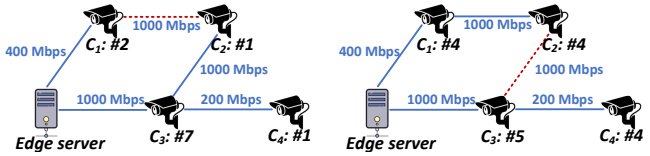
Camera nodes measure real-time link throughput using ongoing video streams and report it to edge server(s) through control plane. Since they take turns to transmit video streams to a relay node (Figure 4) due to medium sharing, throughput measurement window needs to be smaller than the transmission period of each camera node. In Spider, we conduct micro-benchmarks to measure the transmission time of each camera node and empirically set the observation window to 1 ms. The relay node then filters out outlier throughput estimations (below or above 1/4 quartile) and reports the average of remaining measurements to the edge server.

4 Flow planning

This section details Spider's flow planning algorithm. We first describe the flow planning problem (§4.1) and give a formal mathematical formulation (§4.2). We then decouple this problem into application-independent flow routing (§4.3) and application-specific video bit-rate allocation (§4.4), and present our solutions to each of them. We finally describe how our algorithms scale to the multi-edge setting (§4.5).

4.1 Illustrative example: $\max throughput \neq \max accuracy$

While efforts on flow planning [18, 20] dramatically improve the throughput of multi-hop network, they are unsuitable for Spider as maximizing network throughput would not necessarily lead to the maximal video analytics accuracy. Table 1 shows an example video profiling result. Maximizing



(a) Max relay network throughput routing tree.

(b) Max video analytics accuracy routing tree.

Figure 5: Max throughput routing versus max video analytics accuracy routing. The solid line indicates the link is in using.

Cfg. index	1	2	3	4	5	6	7
Bit-rate (Mbps)	12	24	60	120	210	540	900
Accuracy	10%	35%	60%	77%	88%	95%	99%

Table 1: A sample profile from real traffic videos.

the network throughput yields a routing tree (Figure 5(a)) with the following configuration: C_1 transmits at Cfg. #2, C_2 transmits at Cfg. #1, C_3 transmits at Cfg. #7, and C_4 transmits at Cfg. #1. This routing tree and configurations achieve the maximal 948 Mbps overall network throughput and 38.5% overall video analytics accuracy. Maximizing video analytic accuracy, on the other hand, leads to another routing tree (Figure 5(b)) with a different configuration: C_1 transmits at Cfg. #4, C_2 transmits at Cfg. #4, C_3 transmit at Cfg. #5, and C_4 transmits at Cfg. #4, respectively. This routing tree and configuration achieve lower network throughput (570 Mbps) yet a higher (79.75%) average accuracy.

4.2 Problem formulation

We represent mmWave video relay network as a graph $G =< C, L >$, where C and L are the node set and wireless link set, respectively. Each camera node chooses a configuration K to produce a video and forwards it together with video streams from its descendent nodes to the edge server:

$$D_i = \underbrace{T_i(K_i)}_{\text{traffic produced by } C_i} + \underbrace{\sum_{j,R_j=i} D_j}_{\text{traffic produced by the descendent}} \quad (1)$$

where D_i is the video stream to be forwarded by camera node C_i; R_j is the relay node of C_j; $T_i(K_i)$ is the video produced by node C_i at knob K_i. Our goal is to find a proper relay node R_i and configuration K_i for each camera node C_i to maximize the overall video analytics accuracy defined below:

$$\begin{aligned} &\underset{R,K}{\text{maximize}} \quad \sum_{C_i \neq C_{ES}} A_i(K_i)/(|C|) \\ &\text{subject to} \quad U_{C_i} \leq 1,\ C_i \in C. \end{aligned} \quad (2)$$

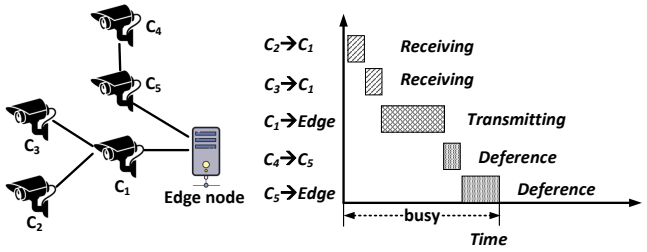


Figure 6: Illustration of radio busy time on camera node C_1.

where $R = \{R_1, R_2, ..., R_{|C|}\}$ and $K = \{K_1, K_2, .., K_{|C|}\}$ are the set of relay variables and configuration variables, respectively. $A_i(K_i)$ is the accuracy of video from camera nodes C_i at configuration K_i. U_{C_i} is the radio utilization of camera node C_i, which should be less than 1. The radio utilization U_{C_i} is defined as the proportion of radio busy time (due to transmitting, receiving, or deference[2] in Figure 6):

$$U_{C_i} = \underbrace{ETT_{l_{(C_i, f(C_i))}} \cdot D_{C_i}}_{\text{transmitting}} + \underbrace{\sum_{C_j \in \{\mathbb{C}(i)\}} ETT_{l_{(C_j, C_i)}} \cdot D_{C_j}}_{\text{receiving}} + \underbrace{\sum_{C_k \in \{\mathbb{S}(i) | C_k \text{not in } \mathbb{C}(i)\}} ETT_{l_{(C_k, f(C_k))}} \cdot D_{C_k}}_{\text{deference}} \tag{3}$$

where $f(C_i)$ is the parent node of C_i; $ETT_{l_{(C_i, f(C_i))}}$ is the Estimated Transmission Time (ETT) [12, 20] of the link between C_i and its parent node $f(C_i)$, which is defined as $ETT_l = 1/\text{Throughput}_l$. $\mathbb{C}(i)$ is the set of children of node C_i. $\mathbb{S}(i)$ is the set of camera nodes in the carrier sensing range of C_i.

Problem relaxation. Enumerating all relay-configuration combinations for each camera node in hopes of finding the optimal solution, however, is computationally intractable. For example, suppose each camera node has 10 video configurations. A relay network with 10 camera nodes leads to a searching space of $10^{10} \times K^{10}$, where K is number of neighbors of each camera node. We decouple the application-independent relay selection (flow routing) from application-dependent configuration selection (video bandwidth allocation) to relax this optimization problem. Specifically, Spider first constructs an optimal routing tree. It then generates a bandwidth allocation plan on this routing tree to maximize the overall video analytics accuracy. The challenge here, however, is to define a proper, application independent objective to guide the routing tree construction.

Spider solves the foregoing challenge based on the key observation that load balance routing improves the overall video analytic accuracy. For instance, a video transmitting at 14 Mbps (210 Mbps for 15 streams) bit-rate yields 88% detection accuracy, as shown in Table 1. Due to the diminishing of video analytics gain with increasing video bit-rate, allocating 22 Mbps extra bit-rate to this video only yields 7% accuracy gain (grows from 88% to 95%). Suppose two camera nodes sharing x Mbps link throughput, a fair bandwidth allocation strategy ($\frac{x}{2}$ Mbps for each camera node) thus tends to yield the maximal average accuracy. Based on this insight, we assign each camera node the same link throughput T and build the routing tree to maximize T.

4.3 Application-independent flow routing

Application-independent flow routing takes the link throughput and medium sharing among nodes as the input and constructs a routing tree rooted at the edge server. Following the insight in previous section, we formulate the flow routing problem below:

$$\begin{aligned} \underset{R}{\text{maximize}} \quad & T \\ \text{subject to} \quad & U_{C(i)} \leq 1,\ C_i \in C. \end{aligned} \tag{4}$$

This optimization problem, unfortunately, is an NP-hard problem. We give the proof in Section A.1. To design an efficient heuristic algorithm, we consider the dual problem of Equation 4. Suppose every camera transmits an unit of data traffic, the dual problem is to minimize the maximum radio utilization in the network.

$$\underset{R}{\text{minimize}} \max_{C_i \in C} U_{C_i} \tag{5}$$

In essence, the bottleneck node should be in the vicinity of the edge server since the upper layer nodes always forward more video streams than the lower layer nodes. This observation gives us two insights. First, the flow routing algorithm should favor those high throughput links since they take less amount of radio time to forward data. Second, we should re-route video streams from the bottleneck node to other low utilization node to alleviate the congestion.

Algorithm. Based on above insights, we design a "generate-and-reroute" two phase flow routing algorithm to minimize the maximal node utilization in the network. To facilitate our presentation, we define the maximal path utilization as the routing path passing through the maximal utilization node.

Our flow routing algorithm (Algorithm 1) first builds a shortest path tree from camera nodes to the edge server using the standard Dijkstra's algorithm ((Line 1, generate phase)). It then alleviates radio utilization of the bottleneck node by relocating cameras along the current maximal utilization path to other routing paths (Line 2-11, reroute phase). In each iteration, the algorithm first re-computes path utilization (Line 3) and then selects a reroute option that leads to the minimal increase of the node ETT (Line 4-9). After that

[2]Deference is the period that the wireless radio can neither receive nor transmit due to carrier sensing.

Algorithm 1: Spider's flow routing

input : 1) Relay network topology $G =< C, L >$
2) Interference map I

output : A flow routing tree R

```
1  R, Cost ← ShortestPathRouting(G);
   // Generate the initial flow routing forest with
      shortest path algorithm
2  for i ← 1 to Threshold do
3      PU ← ComputeUtilization (R, G, I) ;
       // Compute Path Utilization for every node in
          the network
4      for C_p ∈ C do
5          for C_q ∈ C_p.neighbour do
6              if PU[q] < PU[p] then
7                  Δco ← Cost[q] − Cost[p] + ETT(L_{p,q});
8                  UpdateList.append((Δco, p, q));
       // Find possible reroutes to reduce the path
          utilization of congestion link
9      Δco, p, q ← arg min (UpdateList);
                   Δco
10     R_p ← q;
       // Updates the route with least increase in
          routing metric
11     Cost ←UpdateCost(R, G);
```

it updates the routing tree topology and cost (Line 10-11). The algorithm terminates as long as it reaches a pre-defined iteration threshold or there is no path change that leads to a better utilization. The variable *Threshold* in Line 2 is set to the sum of the number of feasible wireless connections from each node, which ensures exploring every possible wireless connection. This algorithm takes $O(L^2)$ iterations to construct the routing tree. We experimentally demonstrate that this heuristic is near-optimal in §6.5.

4.4 Application-specific video bit-rate allocation

This algorithm (re)allocates bandwidth to each camera so as to maximize the overall video analytics accuracy. We replace the variable R in Equation 2 with the routing tree constructed by Algorithm 1, and rewrite Equation 2 as follows:

$$\begin{aligned} &\underset{K}{\text{maximize}} && \sum_{C_i \neq C_{ES}} A_i(K_i)/(|C|) \\ &\text{subject to} && U_{C_i} \leq 1,\ C_i \in C. \end{aligned} \tag{6}$$

Reduction to MILP problem. Finding the optimal allocation plan for all camera nodes, unfortunately, is still a NP-hard combinatorial optimization problem. We give the proof in Section A.2. We thus transform it into a mixed integer linear programming (MILP) problem as follows:

$$\underset{K}{\text{maximize}} \sum_{C_i \neq C_{ES}} \sum_{j} a_{ij} \cdot k_i/(|C|). \tag{7}$$

where k_{ij} is a binary variable that indicates whether camera node C_i chooses configuration j, and a_{ij} is the analytics accuracy when C_i chooses j. This optimization problem should meet the following three conditions.

- **Configuration selection constraint**. Each camera node C_i can only choose one configuration at a time: $\sum_j k_{ij} = 1$. The amount of video traffic produced by a camera node C_i is $t_i = \sum_j t_{ij} \times k_{ij}$, where t_{ij} is the bit-rate of camera node i from configuration j. The video analytic accuracy of the video from this camera is $a_i = \sum_j a_{ij} \times k_{ij}$, where a_{ij} is the accuracy from configuration j.
- **Flow balancing on relay node**. For a given camera node C_i, the total amount of video traffic need to be transmitted by this camera node should equals to the sum of its own video traffic and the video traffic received from its descendant nodes.

$$d_i = t_i + \sum_{j, R_j = i} d_j \tag{8}$$

- **Node utilization condition**. Similarly to the utilization constraint in Equation 6, node utilization should less than 100%.

$$\begin{aligned} ETT_{l_{(C_i, f(C_i))}} \cdot d_i + \sum_{C_j \in \{\mathbb{C}(i)\}} ETT_{l_{(C_j, C_i)}} \cdot d_j \\ + \sum_{C_k \in \{\mathbb{S}(i) | C_k \text{not in } \mathbb{C}(i)\}} ETT_{l_{(C_k, f(C_k))}} \cdot d_k \leq 1 \end{aligned} \tag{9}$$

We then adopt the standard mixed integer linear programming solver [8] to solve the problem. For a network with 200 camera nodes, this solver can find a solution with good approximation in less than one second (§6.5).

4.5 Scaling to multiple edge servers

When multiple edge servers are present in the network, Spider balances both network traffic and compute load among edge servers in both application-independent flow routing and application-specific video bandwidth allocation.

Load balance in flow routing. When there are multiple edge servers distributed over the network, Spider builds a set of routing trees rooted at each edge server with balanced network traffic. To do so, We make the following modifications to Algorithm 1. The first modification is in the generate phase (Line 1 in Algorithm 1), where the algorithm generates M routing trees for M edge servers using the same shortest path algorithm. To balance the size (number of nodes in each trees) of M routing trees, the M edge servers take turns to select the camera nodes into its routing tree based on the

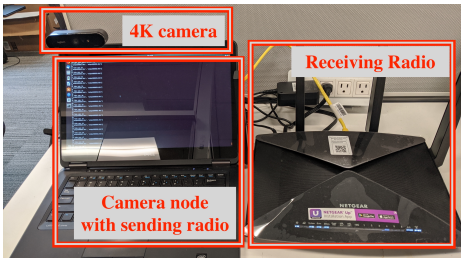
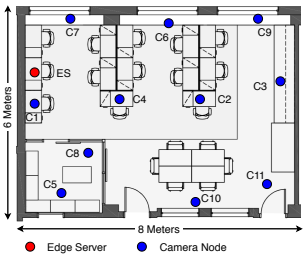


Figure 7: Relay node hardware.

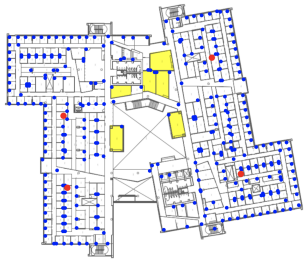
(a) The view of camera node #1.

(b) The view of camera node #7.

Figure 8: Example for camera views.



(a) Deployment of a 11-camera mmWave relay network.



(b) Deployment of a 330-node simulation testbed.

Figure 9: Real-world testbed and trace driven simulation floorplan.

shortest path metric. The node selected by one edge server will not be selected by others. The second modification is in the reroute phase (Line 6), where path utilization is updated within each individual routing tree. These two modifications ensure both intra- and inter-network load balance.

Load balance in video bandwidth allocation. When there are multiple edge servers distributed over the network, the video bandwidth allocation algorithm should also consider the computation capacity on each edge node. To this end, we add a new constraint to the mixed integer linear programming (MILP) problem:

$$\sum_{C_i \in C_{ES_m}} p_{i,j} k_{i,j} \leq P_m \tag{10}$$

where $p_{i,j}$ is the compute resource required for processing video stream from camera C_i with configuration K_j. P_m is the compute capacity of the edge node ES_m.

5 System Implementation

We deploy a Spider testbed in a 6 m × 8 m campus office using 11 camera nodes and one edge server, as shown in Figure 9(a). We include both indoor and outdoor views in camera nodes' deployment as shown in Figure 8(a) and Figure 8(b).

5.1 Testbed description

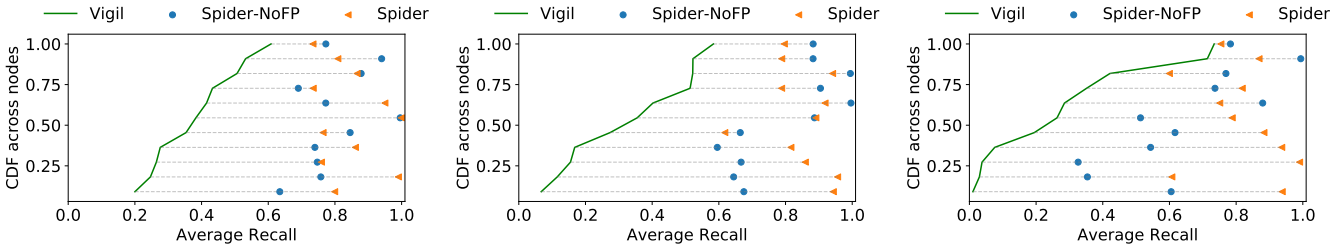
Camera node. Each camera node is a Dell E7440 laptop [4] equipped with a Logitech BRIO 4K camera [6] and two WiGig radios: a QCA6320 mmWave network interface card [10] for transmitting, and a NETGEAR Nighthawk X10 router [9] (802.11ac&ad dual band) for receiving. To reduce the delay on multi-hop video forwarding, each camera node tunnels the incoming video packets (from its child node) by adding its IP address, and forwards it directly to the next-hop receiver. Such a design allows camera node to process incoming video traffic at 2 Gbps using a single CPU core. Video forwarding logic is implemented in C++ to ensure execution efficiency.

Edge server. The edge server is customized from Lambdalabs [5] equipped with a 10-core Intel core i9 CPU, and two NVIDIA RTX 2080 Ti GPUs. It runs efficient cascaded operators for analytics pipelines [7]. Specifically, a background subtraction (BGS) [3] module is first invoked on downsampled frames as an early filter to detect changes in each frame. If this module detects motion in the "region of interest", it calls a DNN model (*e.g.,* YOLOv3 [35] for object detection) executing on the GPU to verify if there is indeed an object of interest. With this cascaded pipelines, Spider is able to process HD videos on one edge server at 370 fps, 45x faster than the baseline of running DNNs on all frames using the same edge server.

5.2 Fault tolerance

Protocols are also implemented in the testbed to deal with link and node failures.

Link failure and throughput degradation. Link failure and throughput degradation are common in mmWave networks and have significant impact on video analytics performance. Spider runs the following protocol to deal with link failure and throughput degradation. The link throughput estimation algorithm (§3.3) detects the abrupt change of throughput locally and reports it to the edge server through control plane straightaway. The edge server then re-run the flow planning heuristics and pushes new receiver ID and video bit-rate configurations (based on bandwidth allocation results) to each camera node through control plane. Once the camera node switches to another link, it periodically checks the availability of the previous (blocked) link using the same back-off idea in CSMA/CD and switches back as soon as the



(a) Object Detection. **(b)** Scene Text Detection. **(c)** Face Detection.

Figure 10: Average recall achieved by three algorithms in different video analytics scenarios. In each figure, the green line is CDF of AR across 11 camera. For each dot in CDF, the corresponding blue and yellow dot is AR achieved by Spider-nofp and Spider for the same camera.

the blocked link is recovered.

Node failure. Once a camera node is unreachable in control plane, the edge server cuts down the wireless links to/from this camera immediately by issuing a request packet to its parent and child nodes. At the same time, the edge server reruns flow planning heuristics and updates routing and camera configurations. After that, the edge server issues a thread to wait for the re-connection (in control plane) request from the failure node.

Edge server failure. Edge server failure has the most severe impact on the system. If there is only one edge server, the failure leads to the stop of the system. When there are multiple edge servers, edge controller follows the same procedure of handling camera node failure and streams videos to other edge nodes. When master edge server fails, it trigger master election in Raft (§3.1). A new master node will be elected and Spider rebuilds flow routing table and configuration information in the new master node.

6 Evaluation

We evaluate Spider using both testbed experiments and trace-driven simulation.

6.1 Experiment setup

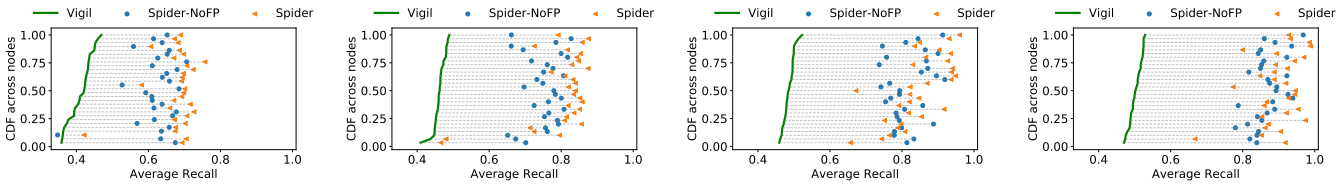
Evaluation methods. We evaluate Spider with three different video analytic tasks: object detection [35], face detection [48], and scene text detection [51]. To emulate large video traffic on our 11 camera node testbed, each camera node transmits five video streams simultaneously. To further understand Spider's performance on enterprise-scale deployment (*e.g.,* cashierless store), we simulate a network topology with 330 camera nodes in a 90 m × 87 m office building, as shown in Figure 9. We estimate peak link throughput at different distances based on the floor plan and signal propagation models of WiGig [45, 52] and Wi-Fi [38, 44], and use video streams collected from our 11-camera testbed.

Baselines. We compare Spider with the following three baseline algorithms. **Vigil** [49]: the state-of-the-art wireless video surveillance system that transmits video streams over 802.11ac Wi-Fi network. **Spider-NoFP**: a naïve version of Spider that adopts shortest path routing without flow planning. **Spider-NC**: a naïve version of Spider that transmits both control packets and video streams over mmWave links.

Ground-truth and video analytics metric. As there is no human-labeled ground-truth on our video streams, we run YOLOv3 on these videos at the highest resolution and take its detection result as the ground-truth. We then define overall video analytics accuracy as the average object detection recall on all network video streams. Recall measures the proportion of the relevant objects being detected, which is critical to many video analytics applications such as cashierless store where higher recall is expected in order to avoid/minimize miss detections. Note that Spider is also compatible with other accuracy metrics (*e.g.,* precision and F1-score).

6.2 End-to-end performance

Single edge server. We compute the average recall (AR) of 2-hours video streams sent from each camera node in the testbed. Figure 10 shows the cumulative distribution function (CDF) of AR for three different video analytic tasks. Each point in the figure corresponds to the AR of one camera node. We observe that AR varies significantly across 11 camera nodes when the network is scheduled by Vigil. The average AR for object detection, scene text detection and face detection is 38.3%, 33.4%, and 28.3%, respectively. In contrast, both Spider-NoFP and Spider outperform Vigil. Spider-NoFP (blue circle) achieves an average AR of 79.8%, 79.9%, and 64.7%, respectively. The uncorrelation between AR of Spider-NoFP and Spider in different nodes demonstrates that the two algorithms have different node preference in flow scheduling. The huge performance gap between Vigil and Spider-NoFP demonstrates the necessity of adopting mmWave links for video transmission. The improvement on average recall is



(a) One Edge Servers. **(b)** Two Edge Servers. **(c)** Three Edge Servers. **(d)** Four Edge Servers.

Figure 11: Average recall achieved by three algorithms in different edge server settings. In each figure, the green line is CDF of AR across 30 edger server locations. For each dot in CDF, the corresponding blue and yellow dot is AR achieved by Spider-nofp and Spider for the same edge server deployment.

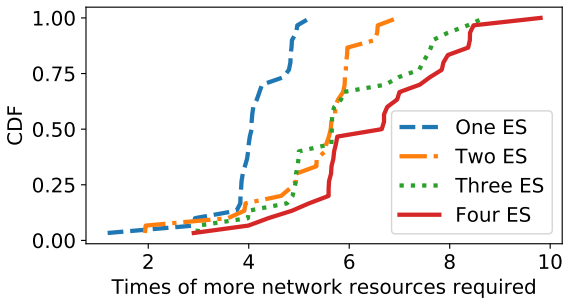
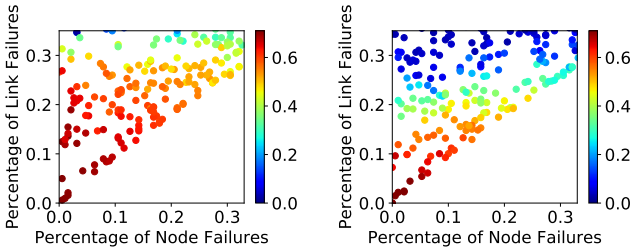


Figure 12: Network resource required for the baseline to achieve the same accuracy as Spider.



(a) Spider. **(b)** Spider-NC.

Figure 13: Impact of node&link failures on average recall.

more significant on fine-grained tasks like face and text detection. With flow planning heuristic, Spider's average AR further grows to 84.2%, 84.7% and 81.2% on these three applications, which is 41.5%–52.9% higher than Vigil. These improvements demonstrate that Spider's flow planning algorithm can effectively improve the video analytics accuracy.

Multiple edge servers. We then test Spider's performance in multiple edge node settings through large-scale simulation. In this experiment, we randomly deploy K=(1,2,3,4) edge servers in the office building and compute AR across all 330 camera nodes. Each experiment is repeated 30 times. Figure 11 shows the CDF of AR in each edge server settings. We observe that Spider achieves consistently better performance than Vigil: 26.5%–39.3%, and 27%–40.8% higher average and median AR, respectively. We also observe that the video analytics accuracy increases with growing number of edge servers. For instance, the mean AR achieved by Spider grows from 67.8% to 79.8% and further to 89.6% as we put one, two, and four edge servers in the network. Such a high performance improvement comes from the load balance mechanism in Spider, which fully exploits the flexibility of relay network and geo-distributed edge nodes to reduce bandwidth and compute contention.

To quantify the benefit of WiGig over Wi-Fi from the network perspective, we consider the scenario where data plane uses 802.11ac links and calculate additional throughput that Vigil demands to achieve the same average AR as Spider. As can be seen in Figure 12, Vigil demands 4.1x, 5.3x, 5.8x, and 6.5x more throughput on average relative to our estimated peak throughput in the simulation environment in one, two, three, and four edge server settings, respectively in our simulated testbed.

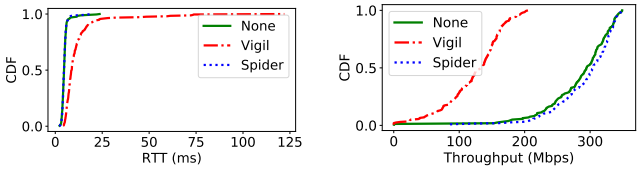
6.3 Roubustness of Spider

Impact of failures. We further run simulations on the 330-node relay network to examine the impact of failures on video analytics performance. We randomly shut down different number of camera nodes and links and compute the average recall of video streams sent from the remaining live camera nodes. For comparison, we also run Spider-NC in the same network topology settings. We show the impact of link and node failures in Figure 13. As expected, the average recall decreases with the growing number of both link and node failures. Specifically, Spider (Figure 13(a)) achieves around 68.0% average recall when there are 10% of dead links and dead nodes in the network. The average recall then drops to around 43.2% when the portion of link&node failures reaches to 30%. In contrast, Spider-NC (Figure 13(b)) achieves 16.6% overall average recall in the same settings. This result demonstrates that Spider's decoupled data plane and control plane improves the reliability of the wireless video analytic system under severe network failure.

Latency of failure recovery. We define link recovery time

Algo.	neighbor disc.	msg. prop.	re-asso.	Total
Spider-NC	5.1s	1.4s	5.0s	11.5s
Spider	n/a	6ms	5.0s	5.0s

Table 2: Time cost for link failure recovery.



(a) network latency **(b)** network throughput

Figure 14: Impact of Spider's control plane on legacy Wi-Fi users.

as the time for the camera node to replace the paralyzed link with the new link that the edge server assigned. We then measure the link recovery time of Spider and Spider-NC. The results are shown in Table 2. Spider-NC takes around 11.5s on average to recovery from a link failure. In contrast, Spider takes around 5s on average, which is 57% reduction in time. This is because Spider-NC conducts beam searching twice for re-association, one for finding a neighbor node that is available to forward its link failure message to the edge server, and another for discovering the new neighbor node allocated by the edge server. In contrast, Spider reports the link failure to the edge server directly through 802.11ac Wi-Fi link. It only searches the beam once to re-associate with the new node that the edge server assigned. Besides, propagating the control packets in the multi-hop mmWave networks also introduce delays due to buffering at relay nodes.

We also compare the delay for updating the routing table at each camera node. For Spider, the camera node takes negligible time to update its routing table since there are reliable, always-on Wi-Fi links between camera nodes and the edge server. Therefore, the camera node can propagate link failure messages to the edge server as soon as the link failure occurs. In contrast, Spider-NC needs to re-associate with an available neighbor node first and then propagate the link failure message to the edge server, which introduces around 6.5s delay on average. These two experiment results demonstrate the overlay network architecture can effectively improve the network response time to link failures.

6.4 Interference to legacy Wi-Fi users

We quantify the impact of Spider control packets on legacy Wi-Fi users in these experiments. We put a legacy Wi-Fi device (a laptop) in our 11-camera node testbed and configure it to work in the same channel as Spider's control plane. This laptop ping a Google server 1,000 times. We then measure

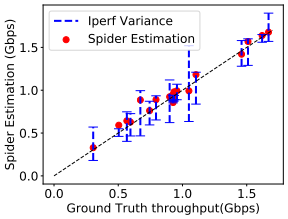


Figure 15: Accuracy for Spider throughput estimation.

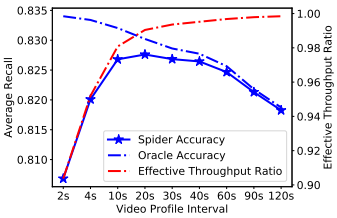


Figure 16: Impact of video profiling interval.

the round trip time (RTT) with and without launching Spider. Figure 14(a) shows the CDF of the RTT measurement. The 95% percentile of RTT is 6.6 ms when there is no control or video streams transmitted over the 802.11ac Wi-Fi band (*i.e.,* neither Vigil nor Spider is working). The 95% percentile of RTT grows slightly to 6.8 ms when Spider transmits control packets over the same Wi-Fi band. This result demonstrates that Spider introduces negligible network delay (0.2 ms) to the legacy Wi-Fi users. In contrast, the 95% percentile of RTT jumps to 23.4 ms when Vigil transmits video streams over the same Wi-Fi band. Running Vigil thus introduces 16.2 ms extra delay, 81x higher than that introduced by Spider.

We further quantify the impact of Spider's control plane on the peak network throughput that legacy Wi-Fi users can achieve. In this experiment, we run iperf3 on the same Wi-Fi device and record the uplink throughput every one second. Figure 14(b) shows the CDF of network throughput. We observe that this legacy Wi-Fi device can achieve almost the same network throughput no matter Spider is running or not. In contrast, the network throughput drops significantly when video streams are transmitted all through 802.11ac Wi-Fi links (Vigil). Specifically, the 95% percentile of the maximum network throughput achieved by the legacy Wi-Fi device is 305.0 Mbps. However, the maximum network throughput drops to around 133.5 Mbps when Vigil is running.

6.5 Micro-benchmark

Lastly, we conduct micro-benchmarks to understand the performance of each functional module in Spider.

Accuracy of throughput estimation. In this experiment, we evaluate the link throughput estimation algorithm in different link throughput settings. We use the throughput reported by iPerf3 as the groundtruth. Figure 15 shows the result. Each dot in the figure represents a link throughput estimation. We observe Spider achieves consistently high throughput estimation accuracy in both low ($\leq$ 1 Gbps) and

high (> 1 Gbps) link throughput settings. All these throughput estimation results are within the error range of throughput estimation reported by iperf3.

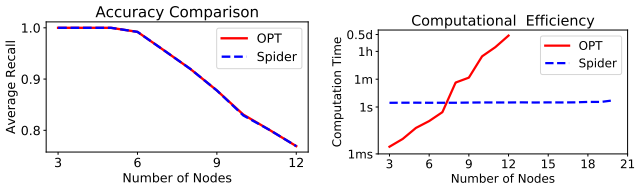
Impact of video profiling. Each camera node in Spider periodically transmits video segments to the edge server over the data plane. We vary the video profiling interval to investigate its impact on the overall video analytics accuracy. To facilitate our presentation, we define effective throughput ratio as the ratio between the non-profiling video traffic and the overall video traffic transmitted over the data plane. Short video profiling interval would yield a more accurate resource-accuracy profile, at the cost of a larger amount of profiling traffic (*i.e.,* lower effective throughput ratio). For comparison, we assume there is an oracle transmitting video profiling samples on a different channel (*i.e.,* does not share the mmWave links with normal video streams). The result is shown in Figure16. Spider's average recall grows rapidly as we increase the video profiling interval from 2 s to 20 s. This is because the bandwidth allocated to non-profiling video streams grows with increasing of video profiling interval. Nonetheless, Spider fails to capture temporal variations of video content as the profiling interval grows further. Accordingly, we observe the average recall begins to drop at 20 s interval settings. Spider achieves 82.7% average recall in 20 s interval settings, 0.5% lower than the oracle. At the same time, the effective throughput ratio maintains in a high level (>99%). Suggested by this result, we choose 20 s as the default video profiling interval in our design.

Optimality and efficiency of the flow planning algorithm. We first compare the average recall achieved by Spider with the maximal average recall achieved by brute-force search algorithm. As the brute-force search algorithm (OPT) is computationally intractable, we only obtain the maximal average recall in small network settings (≤12 camera nodes). The result is shown in Figure 17(a). We observe Spider achieves almost the same average recall as the brute-force search algorithm, demonstrating the effectiveness of our flow planning heuristics.

We next compare the computation efficiency of Spider's flow planning heuristic with brute-force search algorithm (OPT). The result is shown in Figure 17(b). We observe Spider's heuristic takes consistently short time (*i.e.,* less than 2 s). In contrast, computation cost of brute-force search algorithm grows exponentially from 5 ms to 1 min, and 12 hours as we increase the number of camera node from 3 to 9, and further to 12.

7 Related Work

Spider builds on a long tradition of network optimization and upon recent advances in computer vision, machine learning, and wireless communication.



(a) The optimality of Spider. **(b)** The efficiency of Spider.

Figure 17: The optimality and efficiency of Spider flow planning algorithm.

Video analytics systems. Live video analytics is becoming more pervasive due to the increasing number of cameras and the deployment of edge computing nodes [24–27, 46, 47, 49]. However, the network resource is one of the key constraints in the design of those system. Vigil [49] limits the frame resolution and transmitting objects due to the limited network bandwidth. Zero-stream [46] sacrifices the timeliness of video to reduce bandwidth usage. Different from those works, Spider demonstrates high bandwidth network with application-oriented flow scheduling can fully unleash the potential of video analytics system.

mmWave network. Mm-wave communication is emerging as one of the key technologies in the 5G era. Efforts have been spent in the research community to model mmWave networks [52], improve mmWave link coverage [41–43], reduce beam alignment and AP switching delay [23, 34, 50], and build multi-radio hybrid network with Wi-Fi or LTE [39]. mmWave has also been used to stream videos and support applications like VR/AR that demand high bandwidth and low latency [16]. Spider builds on top of these techniques, and we view it is orthogonal to the mmWave advancements below the transport layer.

Multi-hop networks. Multi-hop relay technologies have been extensively studied in wireless mesh networks [17, 18, 20, 29]. There is also relay network design based on 60 GHz radios: Terragraph [11]. Unlike the focus of Terragraph towards optimization network throughput, Spider focuses on maximizing video analytics accuracy through a joint optimization of network flow and application configuration. In addition, different from Terragraph, Spider takes a different approach in network architecture design, which separates dedicate control plane for network control.

8 Conclusion

We have built Spider, a live video analytics system based on multi-hop, mmWave relay network. With the objective of maximizing the overall video analytics accuracy, Spider proposes to integrate a separate low-latency Wi-Fi control plane

with the high-throughput mmWave data plane, which allows the edge server to timely schedule the network routing and video bit-rate allocation. We have implemented a prototype of Spider and conduct both real-world experiments and large-scale simulations. Results show that Spider dramatically improves video analytics accuracy, improves system robustness, and reduces interference to existing users.

References

[1] AWS Outposts. https://aws.amazon.com/outposts/.

[2] Azure Stack Edge. https://azure.microsoft.com/en-us/services/databox/edge/.

[3] Background Subtraction Methods - OpenCV. https://docs.opencv.org/3.4/d1/dc5/tutorial_background_subtraction.html.

[4] Dell E7440. www.newegg.com.

[5] Lambda Labs. https://lambdalabs.com/.

[6] Logitech brio webcam with 4k ultra hd video. NETGEAR.

[7] Microsoft Rocket Video Analytics Platform. https://github.com/microsoft/Microsoft-Rocket-Video-Analytics-Platform.

[8] MILP Solver. https://developers.google.com/optimization/mip/integer_opt.

[9] Netgear nighthawk x10 router. NETGEAR.

[10] Ngff595a-l, qualcomm atheros qca6320-based 802.11ad wigig + qca6174-based 802.11a/b/g/n/ac + 4.1 bt m.2 module. NGFF595A-L.

[11] Terragraph: Solving the Urban Bandwidth Challenge. https://terragraph.com/.

[12] D. Aguayo, J. Bicket, R. Morris. Srcrr: A high throughput routing protocol for 802.11 mesh networks (draft). *MIT, Tech. Rep*, 2005.

[13] Amazon Go. https://www.amazon.com/b?ie=UTF8&node=16008589011.

[14] Inside Amazon Go: The camera-filled convenience store that watches you back. The Washington Post.

[15] G. Ananthanarayanan, V. Bahl, Y. Shu, F. Loewenherz, D. Lai, D. Akers, P. Cao, F. Xia, J. Zhang, A. Song. Traffic Video Analytics – Case Study Report. Tech. rep., Microsoft and City of Bellevue, 2019.

[16] G. Baig, J. He, M. A. Qureshi, L. Qiu, G. Chen, P. Chen, Y. Hu. Jigsaw: Robust live 4k video streaming, 2019.

[17] J. Bicket, D. Aguayo, S. Biswas, R. Morris. Architecture and evaluation of an unplanned 802.11 b mesh network. *Proceedings of the 11th annual international conference on Mobile computing and networking*, 31–42, 2005.

[18] S. Biswas, R. Morris. Opportunistic Routing in Multi-Hop Wireless Networks. *SIGCOMM Comput. Commun. Rev.*, **34**(1), 69–74, 2004.

[19] Cisco. IP Video Surveillance Design Guide. https://www.cisco.com/c/en/us/td/docs/solutions/Enterprise/Video/IPVS/IPVS_DG/IPVS-DesignGuide/IPVSchap4.html#Number_of_Cameras_per_Location.

[20] R. Draves, J. Padhye, B. Zill. *ACM MobiCom*, 2004.

[21] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, P. Levis. Collection tree protocol. *ACM Sensys*, 2009.

[22] M. K. Haider, Y. Ghasempour, D. Koutsonikolas, E. W. Knightly. Listeer: Mmwave beam acquisition and steering by tracking indicator leds on wireless aps. *ACM MobiCom*, 2018.

[23] H. Hassanieh, O. Abari, M. Rodriguez, M. Abdelghany, D. Katabi, P. Indyk. Fast millimeter wave beam alignment. *ACM SIGCOMM*, 2018.

[24] K. Hsieh, G. Ananthanarayanan, P. Bodik, S. Venkataraman, P. Bahl, M. Philipose, P. B. Gibbons, O. Mutlu. Focus: Querying large video datasets with low latency and low cost. *USENIX OSDI*, 2018.

[25] A. H. Jiang, D. L.-K. Wong, C. Canel, L. Tang, I. Misra, M. Kaminsky, M. A. Kozuch, P. Pillai, D. G. Andersen, G. R. Ganger. Mainstream: Dynamic stem-sharing for multi-tenant video processing. *USENIX ATC*, 2018.

[26] J. Jiang, G. Ananthanarayanan, P. Bodik, S. Sen, I. Stoica. Chameleon: scalable adaptation of video analytics. *ACM SIGCOMM*, 2018.

[27] D. Kang, J. Emmons, F. Abuzaid, P. Bailis, M. Zaharia. Noscope: optimizing neural network queries over video at scale. *Proceedings of the VLDB Endowment*, 2017.

[28] R. M. Karp. Reducibility among combinatorial problems. *Complexity of computer computations*, 85–103. Springer, 1972.

[29] S. Lee, B. Bhattacharjee, S. Banerjee. Efficient Geographic Routing in Multihop Wireless Networks. *ACM MobiHoc*, 2005.

[30] M. H. Mazaheri, S. Ameli, A. Abedi, O. Abari. A millimeter wave network for billions of things. *Proceedings of the ACM Special Interest Group on Data Communication*, 2019.

[31] Multiprocessor scheduling problem. https://onlinelibrary.wiley.com/doi/10.1002/0471727253.ch9.

[32] The NBA's Fastbreak into Video Analytics. https://prism.com/blog/2015/06/16/nba-fastbreak-video-analytics/.

[33] D. Ongaro, J. Ousterhout. In search of an understandable consensus algorithm. *USENIX Annual Technical Conference*, 305–319, 2014.

[34] J. Palacios, D. Steinmetzer, A. Loch, M. Hollick, J. Widmer. Adaptive codebook optimization for beam training on off-the-shelf ieee 802.11 ad devices. *ACM MobiCom*, 2018.

[35] J. Redmon, A. Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[36] V. Růžička, F. Franchetti. Fast and accurate object detection in high resolution 4K and 8K video using GPUs. *IEEE HPEC*, 2018.

[37] R. Singh, P. R. Kumar. Throughput Optimal Decentralized Scheduling of Multihop Networks With End-to-End Deadline Constraints: Unreliable Links. *IEEE Transactions on Automatic Control*, **64**(1), 127–142, 2019.

[38] W. C. Stone. Electromagnetic signal attenuation in construction materials. Tech. rep., National Institute of Standards and Technology, 1997.

[39] S. Sur, I. Pefkianakis, X. Zhang, K.-H. Kim. Wifi-assisted 60 ghz wireless networks. *ACM MobiCom*, 2017.

[40] S. Sur, V. Venkateswaran, X. Zhang, P. Ramanathan. 60 ghz indoor networking through flexible beams: A link-level profiling. *ACM SIGMETRICS Performance Evaluation Review*, vol. 43, 71–84. ACM, 2015.

[41] S. Sur, X. Zhang, P. Ramanathan, R. Chandra. Beamspy: enabling robust 60 ghz links under blockage. *NSDI*, 2016.

[42] T. Wei, X. Zhang. Pose information assisted 60 ghz networks: Towards seamless coverage and mobility support. *ACM MobiCom*, 2017.

[43] T. Wei, A. Zhou, X. Zhang. Facilitating robust 60 ghz network deployment by sensing ambient reflectors. *NSDI*, 2017.

[44] R. Wilson. Propagation losses through common building materials 2.4 ghz vs 5 ghz. *Magis Networks Inc.: San Diego, CA, USA*, 2002.

[45] J. C. Wiltse. Corrections to published curves for atmospheric attenuation in the 10 to 1000 ghz region. *IEEE Antennas and Propagation Society International Symposium 1997. Digest*, vol. 4, 2580–2583. IEEE, 1997.

[46] M. Xu, T. Xu, Y. Liu, X. Liu, G. Huang, F. X. Lin. Supporting video queries on zero-streaming cameras. *arXiv preprint arXiv:1904.12342*, 2019.

[47] H. Zhang, G. Ananthanarayanan, P. Bodik, M. Philipose, P. Bahl, M. J. Freedman. Live video analytics at scale with approximation and delay-tolerance. *USENIX NSDI*, 2017.

[48] K. Zhang, Z. Zhang, Z. Li, Y. Qiao. Joint face detection and alignment using multitask cascaded convolutional networks. *IEEE Signal Processing Letters*, **23**(10), 1499–1503, 2016.

[49] T. Zhang, A. Chowdhery, P. V. Bahl, K. Jamieson, S. Banerjee. The design and implementation of a wireless video surveillance system. *ACM MobiCom*, 2015.

[50] A. Zhou, L. Wu, S. Xu, H. Ma, T. Wei, X. Zhang. Following the shadow: Agile 3-d beam-steering for 60 ghz wireless networks. *IEEE INFOCOM*, 2018.

[51] X. Zhou, C. Yao, H. Wen, Y. Wang, S. Zhou, W. He, J. Liang. East: an efficient and accurate scene text detector. *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 5551–5560, 2017.

[52] Y. Zhu, Z. Zhang, Z. Marzi, C. Nelson, U. Madhow, B. Y. Zhao, H. Zheng. Demystifying 60ghz outdoor picocells. *ACM MobiCom*, 2014.

A Appendix

A.1 NP-hard reduction of flow routing problem

We reduce multiprocessor scheduling problem [31], a well-known NP-hard problem to our flow routing problem.

Given m processors and J jobs where each job j_i takes l_i time for execution, the multiprocessor scheduling problem aims to find the minimum possible time required to schedule all jobs in J on m processors such that none of two jobs overlap with each other. We reduce multiprocessor scheduling problem to our flow routing problem as follows: we first construct a network with m edge servers and $|J|$ camera nodes. We then define the cost of a wireless link between an edge server and a camera node i as l_i/H, where H is a large constant number to ensure $l_i/H < 1$. This is the simplified version of our flow routing problem where no wireless interference among parallel transmissions. Solving this flow routing problem in polynomial time contradicts to the NP-hardness of multiprocessor scheduling problem. Hence Spider's flow routing problem is NP-hard.

A.2 NP-hard reduction of Video Bit-rate Allocation

We prove the NP-hardness by reducing the Knapsack problem, another well-known NP-hard problem [28] to our video bit-rate allocation problem.

Consider a 0-1 knapsack problem that has n objects. Each object i is associated with a weight w_i and a value v_i. The goal is of 0-1 knapsack problem is to maximize $\sum_{i=1}^{n} v_i x_i$ with the constraints that $\sum_{i=1}^{n} w_i x_i \leq W$. We reduce the above problem to our video bit-rate allocation problem as follows: we first build a network with one edge server and n camera nodes with each camera node directly connects to the edge server. We then define the accuracy and bandwidth requirement of this node as: v_i/v_{max} and w_i/W, where v_{max} and W are constant values to ensure $v_i/v_{max} < 1$ and $w_i/W < 1$. This is the simplified version of our video bit-rate allocation problem where each node either transmits or not. Solving this problem in polynomial time would contradict with the NP-hardness of the 0-1 knapsack problem. Hence Spider's video bit-rate allocation problem is NP-hard.