Spider: A Multi-Hop Millimeter-Wave Network for Live Video Analytics

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

[1] Princeton University, [2] Microsoft

Abstract

Massive video analytics systems, comprised of many densely-deployed cameras and supporting edge servers, are driving innovation in many areas including smart retail stores and security monitoring. To support such systems the challenge lies in collecting video footage in a way that maximizes end-to-end application goals, and scales this performance as camera density increases to meet application needs. This paper presents *Spider*, a multi-hop, millimeter-wave (mmWave) wireless relay network design that meets these needs. To mitigate physical mmWave link blockage, Spider integrates a low-latency Wi-Fi control plane with a mmWave relay data plane, allowing agile re-routing around blockages. Spider proposes a novel video bit-rate allocation algorithm coupled with a scalable routing algorithm that works hand-in-hand toward the application-level objective of maximizing video analytics accuracy, rather than simply maximizing data throughput. Our experimental evaluation uses a combination of testbed deployment and trace-driven simulation and compares against both Wi-Fi and mmWave mesh schemes that operate without Spider's algorithms. Results show that Spider is able to support camera densities up to 176% higher (gains of 2.76×) than the best-performing comparison scheme, allowing it alone to meet real-world camera density targets (4–250 cameras/1,000 sq. ft., depending on application). Further experiments demonstrate Spider's scalability in the presence of failures, with a 5.4–100× reduction in average failure recovery time.

ACM Reference Format:
Zhuqi Li, Yuanchao Shu, Ganesh Ananthanarayanan, Longfei Shangguan, Kyle Jamieson, Paramvir Bahl. 2021. Spider: A Multi-Hop Millimeter-Wave Network for Live Video Analytics: . In *The Sixth ACM/IEEE Symposium on Edge Computing (SEC '21), December 14–17, 2021, San Jose, CA, USA*. ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3453142.3491291

SEC '21, December 14–17, 2021, San Jose, CA, USA

ACM ISBN 978-1-4503-8390-5/21/12…$15.00
https://doi.org/10.1145/3453142.3491291

1 Introduction

Live video analytics is central to many emerging applications like cashierless store management [3] and security monitoring [41, 42], where hundreds of cameras work together to monitor regions of interest [24] and cameras stream their video feeds to on-premise edge servers [6, 7] equipped with accelerators (*i.e.,* GPUs) to execute machine learning models like DNN-based object detection and tracking. The performance of these machine learning models is highly sensitive to variations in video content and quality [61]. Applications' needs for high video analytics accuracy thus drive a technological push for higher resolution and video frame rates: prior work shows up to a 2.2× improvement in accuracy for object detection with 4K video resolution and frame rates of 60 frames per second [47].

Applications also often require a dense, large-scale camera network to achieve seamless coverage. For example, cashierless stores use one camera per four square feet (estimated from photos of an actual Amazon Go store [2]). The area of a typical small cashierless store ranges from 1,200–2,700 square feet, thus requiring 300–675 cameras. This in turn requires tens of Gbits/second bandwidth in the aggregate, a requirement that outclasses both Wi-Fi and sub-6 GHz cellular systems by an order of magnitude. While connecting massive video camera deployments via wired backhaul is possible, it incurs high labor and construction cost to route cables from distributed cameras to edge servers. In addition, the cost goes up when we scale the deployment to a larger area (e.g., enterprise building) where much longer cables are needed.

Millimeter-wave (mmWave) wireless networks, with multi-Gbit/second capacities (*e.g.,* 802.11ad WiGig [40] at 60 GHz), are a promising high bit-rate alternative at a reasonable cost, as the price of a mmWave radio is now roughly the same as a Wi-Fi radio (15 USD) [5], and do not require backbone cables and thus save most of the material and labor cost. However, they do have well-documented limitations: (*i*) their high throughputs depend on clear line-of-sight, thus making them susceptible to moving objects and people in the building, and (*ii*) their connection range is shorter than typical Sub-6 GHz communication technologies.

Spider: A mmWave multi-hop network design for video analytics. We propose *Spider*, a mmWave *multi-hop relay* network to support densely-deployed live video analytics systems. As shown in Fig. 1, Spider fits WiGig radios to edge servers, while cameras relay their video streams to the edge

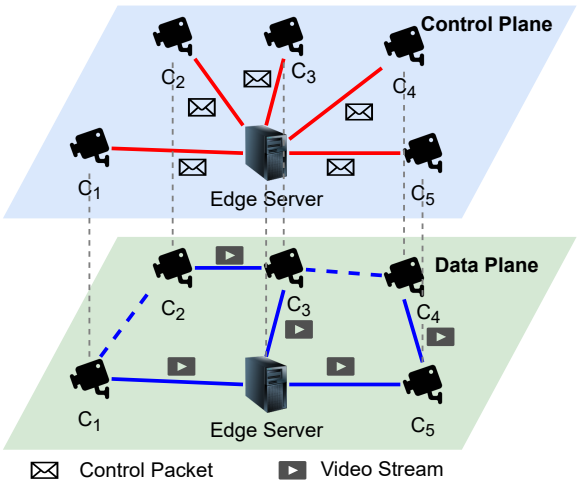


Fig. 1— Spider's data and control flows. High volume video streams traverse mmWave links (solid lines in the data plane). Short control packets are forwarded over long-range wireless links (solid lines in the control plane).

server for analytics processing. Cameras close to the edge servers communicate directly, while farther-away cameras pass their streams to other relay cameras that in turn forward them to an edge server. As a result, network links of relay cameras closer to the edge server will experience higher utilization and congestion.

While there have been extensive studies on building single-hop mmWave networks [19, 20, 36, 57, 64], no effort has yet been focused on large-scale video streaming over multi-hop mmWave networks. Constructing such a mmWave relay network is difficult because if one link in such a network becomes disconnected as a result of physical blockage [9, 66], the narrow beamwidth of that link means that the radio endpoints would then have to search for a possible alternate reflection path or alternate route in order to reconnect the network. This is problematic because the new relay node 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 [20, 53, 66]. Furthermore, the necessary dissemination of routing information to each node's neighbors in the mmWave relay network is challenging, since each radio has to beam towards each of its neighbors, which is costly in time. The conclusion is that we require a separate *control plane* to enable rapid, centralized routing information collection and dissemination, which we describe in §2.1.

Flow routing in multi-hop relay networks is a well-studied problem [12, 16, 18, 33, 51]. However, Spider's design objective is fundamentally different, as it focuses on the application-level objective of *video analytics accuracy*: unlike prior work, our routing protocol does *not* optimize for sum network flow throughput [48], but instead alleviates the congestion of bottleneck nodes by taking into account the potential interference between directional beams and dynamics in link throughput. As a result, it often selects routes that might be sub-optimal in terms of sum network flow, but lead to less congestion in the network and thus allocates each camera sufficient network resources to transmit its stream at good video-analytic accuracy. We describe Spider's routing protocol in §2.2.

For the purpose of building the Spider network, multiple streams need to share a single mmWave link, and each such link is highly dynamic [9], carrying varying amounts of video stream content [26]. This motivates us to develop a multi-camera video bit rate allocation algorithm (*Spider-VBA*, §2.3) to adjust each camera's offered encoded video bit-rate and to allocate each camera's video stream to each network link, in a way that maximizes video analytics accuracy. Spider-VBA formulates the complex, non-linear bandwidth-accuracy tradeoff, which also varies across cameras and with time, as an optimization problem, and solves it using a *mixed integer linear programming* (MILP) model. It then disseminates video bit-rates through the control plane to all cameras and relays in the network.

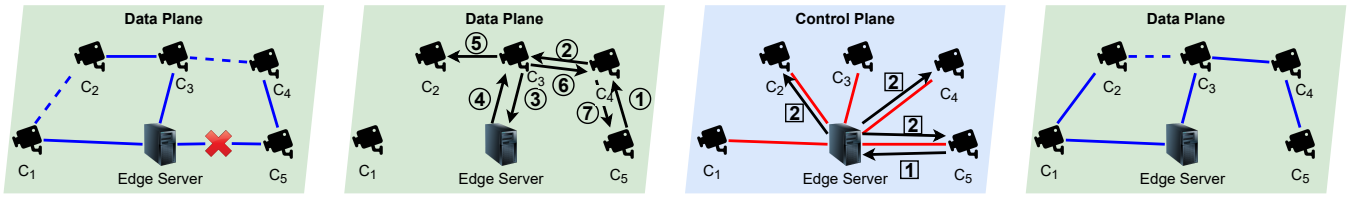
We have built a Spider prototype in a building with 11 distributed camera nodes. We conduct experiments and trace-driven simulations in our the testbed. Results show that Spider can support the camera network with up to 2.76× greater deployment density compared to baseline Wi-Fi and mmWave systems that do leverage neither Spider-VBA nor Spider's decoupled control and data plane architecture. Further experiments demonstrate Spider's scalability in the presence of failures, with a 5.4–100× reduction in average failure recovery time against the same set of baselines.

2 Design

Spider leverages high-throughput mmWave links to build a high-accuracy and scalable wireless video analytic system, with the objective of *maximizing overall live video analytic accuracy*. To this end, Spider makes judicious decisions on *i*) the amount of data produced by each camera, and *ii*) data routing in the multi-hop mmWave relay network. Spider decouples the control plane from the data plane in the frequency domain, which allows a rapid collection of routing-related control packets over the lower-frequency band, and an unobtrusive bulky video data dissemination over the mmWave relay networks. (§2.1). To optimize the routing selection for bulky video data transfer, Spider's routing algorithm exploits the directionality of mmWave links to parallelize concurrent flows with minimal *path congestion* (§2.2). In the application layer, Spider incorporates a flexible, application-specific video bit-rate allocation algorithm (Spider-VBA) to dynamically configure video bit-rate for each camera so that the overall video analytic accuracy is maximized (§2.3).

Spider contains two types of devices.

Camera node. Camera nodes produce live video streams and



(a) Topology of a mmWave relay network. **(b)** Re-routing steps using the data plane. **(c)** Re-routing using the control plane. **(d)** Network topology after re-routing.

Fig. 2— Re-routing after a link failure with/without a control plane. Solid lines in the data plane are routes in mmWave relay network and dashed lines are alternative routes. Solid lines in the control plane are Wi-Fi links for the collection and dissemination of routing information. Arrows represent failure recovery steps.

also forward other cameras' video streams through mmWave links to the edge server. In addition, each camera produces high-resolution video samples periodically and sends them to the edge server for resource-accuracy profiling. To facilitate network management, camera nodes also periodically report the data rate of all associated links to the edge server.

Edge server. Edge server feeds video streams directly to its *analytics engine* for video content analysis. To deal with content variations and link throughput dynamics, edge server periodically profiles high bit-rate video samples from each camera and updates its resource-accuracy profile. Based on the latest resource-accuracy profile and link status reports, edge server runs a reactive scheduler to route network flows and reallocate bandwidth to each camera.

2.1 Network Architecture

Spider adopts a multi-hop relay architecture to expand the network to a large group of cameras while benefiting from the high throughput of mmWave links. However, relay architecture poses significant challenges to data routing as mmWave links are highly directional and susceptible to blockage.

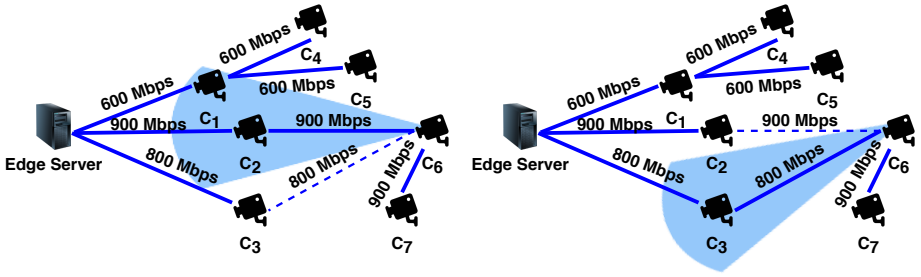
In a mmWave relay network, a node ought to associate and update network status (*e.g.,* link and node failures) with its neighbors separately due to the narrow and directional beam of its antenna arrays. This is a time-consuming process as setting up a mmWave link alone would take up to 1.5 s [20]. Therefore, propagating a network status update to all the nodes in a relay network with hundreds of camera nodes could take 30-60 seconds. In addition, broadcasting routing information would inherently pause the on-going data forwarding on the parent node as the former has a higher priority. As a result, it could lead to high application layer latency and link failure rate due to link status changes during route request flooding.

Fig. 2(a) shows how new routes are established after a link failure in mmWave relay network. When the link between C_5 and edge server breaks, the failure information has to go through C_4 (Step ①) and C_3 (Step ②), all through their mmWave channels, to the edge server (as shown in Fig. 2(b)). Specifically, in order to establish a connection between C_4 and C_3, C_3 has to pause its ongoing video forwarding and undertakes beam searching to align with C_4. Such link failure recovery can be significantly slow in a large network as link establishment involves time-consuming beam searching.

An intuitive solution is to pre-compute the alternative routes and cache these backup routing plans locally at each camera node. Such design, however, runs into problems due to the following issues. First, the impact of a single link failure may spread over the entire network. We take the same topology shown in Fig. 2(a) as an example. When both C_4 and C_5 have to route their traffic through C_3 due to the link failure between C_5 and the edge server, C_2 has to re-route its traffic as well to avoid saturating the link between C_3 and edge server. As C_2 is not aware of the link failure between C_5 and the edge node, C_5 has to notify C_2 through multiple hops, which inevitably introduces significant delay. Second, the combination of possible link failures grows exponentially with the network size. Accordingly, computing the alternative routing plan for each link failure case tends to be computationally intractable.

Decoupling the control and data planes. Spider introduces a dedicated control plane at a lower frequency (*e.g.,* Wi-Fi) that can communicate over a much longer distance and is more robust to the presence of objects and people in indoor environments. Cameras send link status updates through Spider's control plane. The edge server collects these updates, making centralized routing decisions, and then disseminates routes back to the cameras via the control plane. In the example of Fig. 2(a), once the link between camera C_5 and the edge node breaks, instead of propagating this issue through the data plane, C_5 reports the link failure to the edge server directly through the control plane(Step 1), as shown in Fig. 2(c). The edge server then makes a centralized re-routing decision, and notifies C_5 as well as C_2 and C_4 to update their routes, through the control plane as well. Upon receiving the new configuration, these three cameras take a one-shot beam searching to switch to their new parent nodes concurrently, thereby minimizing link failure recovery latency. With such a design, the control plane at the lower frequency can help the mmWave link recover from failure immediately, which allows the multi-hop mmWave network deliver the video streams robustly.

In what follows, we use 802.11ac Wi-Fi as the control plane (upper blue plane) and 802.11ad WiGig mmWave as



(a) Max-flow routing. **(b)** Spider routing.

Fig. 3— Max-flow routing versus Spider routing. Solid lines indicate the link is in use. Shadow area is the beam for Camera C_6. Spider adjusts the interference pattern of mmWave radios to achieve maximal flow parallelism.

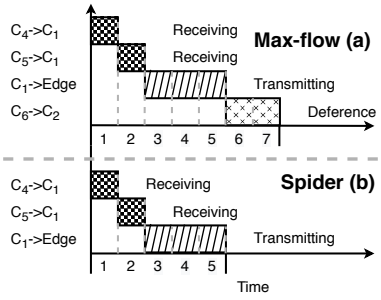


Fig. 4— Transmission timeline for max-flow and Spider routing (*cf.* Fig. 3), respectively.

the data plane (lower green plane) in Spider's design (Fig. 1).[1] Overall, Spider produces on average only ten Kbit/s control traffic over the entire relay network. Such a small amount of traffic has a negligible effect on legacy users in the same Wi-Fi/LTE band, as we experimentally demonstrate in §4.4.1.

2.2 Routing algorithm

Spider's routing algorithm has to cope with a collection traffic pattern from camera nodes to the edge server. However, uncoordinated concurrent flows (video streams) tend to interfere with each other and create bottlenecks in the mmWave relay network [4]. As illustrated in Fig. 3(a), the transmission from Camera C_6 to C_2 causes interference at C_1.

Figures 3 and 4 demonstrate the impact of wireless medium sharing on flow parallelism. As Fig. 4(a) shows, Camera C_1 in Fig. 3(a) needs to divide time slots into seven parts, two of which to receive video streams from C_4 and C_5, three to transmit data (*i.e.,* flows originating from Camera nodes C_1, C_4, and C_5) to the edge server, and another two to wait for the transmission from C_6 to the edge server. The seven time slots cannot overlap each other in order to avoid interference in the shared medium.

Strawman: Max-flow routing. A natural idea for Spider could be to formulate data forwarding as a maximum flow problem [48], but this does not take interference and relay considerations into account. As the example in Fig. 3(a) shows, Max-flow routing prefers the route $C_6 \rightarrow C_2 \rightarrow$ ES to the route $C_6 \rightarrow C_3 \rightarrow$ ES, due to the higher link throughput in the former path. However, route $C_6 \rightarrow C_2 \rightarrow$ ES leads to more flows crowded at C_1. As Fig. 4(a) shows, there are seven non-overlapped time slots at C_1.

As opposed to Max-flow routing, Spider's routing algorithm focuses on increasing the parallelism of concurrent flows by exploiting mmWave's favorable spatial reuse properties. As a result, it leads to less congestion in the network and thus allows each camera node to get more time slots to transit its video data. As shown in Fig. 3(b), C_6 chooses the route $C_6 \rightarrow C_3 \rightarrow$ ES and thus routes the traffic away from the bottleneck. The resultant number of non-overlapped time slots at C_1 hence reduces to five (Fig. 4(b)).

In addition to flow parallelism, link throughput also affects network congestion. A higher link throughput means more data can be transmitted given the same time slot and therefore leads to less congestion. To model the effect of both flow parallelism and link throughput, we formally define the *congestion* of a node C_i as:

$$W_{C_i} = \underbrace{\underset{(C_i,p(C_i))}{ETT} N_{C_i}}_{\text{transmitting}} + \underbrace{\sum_{C_j \in \{\mathbb{C}(i)\}} \underset{(C_j,C_i)}{ETT} N_{C_j}}_{\text{receiving}} + \underbrace{\sum_{C_k \in \{\mathbb{I}(i)\}} \underset{(C_k,p(C_k))}{ETT} N_{C_k}}_{\text{deference}} \quad (1)$$

where $p(C_i)$ is the parent node of C_i, $\mathbb{C}(i)$ is the set of children of node C_i, $\mathbb{I}(i)$ is the set of camera nodes that C_i can overhear their transmission, N_{C_i} is the number of flows transmitted by camera node C_i, and $\underset{(C_i,C_j)}{ETT}$ is the Estimated Transmission Time (ETT) [1, 16] of the link between C_i and node C_j, which is defined as the inverse of link throughput.

Node congestion represents the number of non-overlapped time slots, weighted by link throughput. Non-overlapped time slots classify into three types: transmitting, receiving, and deference[2]. The bottleneck of a relay network is the node with maximum congestion. When the congestion of bottleneck nodes decrease, camera streams get more network resources to transmit at a higher bit-rate. Therefore, Spider's routing algorithm aims to reduce the congestion of bottleneck nodes by picking up the routing plan R:

$$\underset{R}{\text{minimize}} \max_{C_i \in C} W_{C_i} \quad (2)$$

As shown in Equation 1, two major factors affect node congestion. The first is link throughput. A higher throughput link transmits more data given a certain amount of time. The second is non-overlapped time slots. Fewer non-overlapped time slots mean more time can be allocated to each time slot. With these two insights, Spider's routing algorithm routes flows away from bottleneck nodes in the network by searching

[1] Likewise, LTE could be the control plane for larger coverage.

[2] Deference flows are those a wireless radio can hear 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      PW ← ComputeCongestion(R, G, I);
       // Compute Path Congestion for every
          node in the network
4      for C_p ∈ C do
5          for C_q ∈ C_p.neighbor do
6              if PW[q] < PW[p] then
7                  Δco ← Cost[q] − Cost[p] + ETT_(C_p,C_q) ;
8                  UpdateList.append((Δco, p, q));
       // Find possible reroutes to reduce
          the path congestion of bottleneck
          link
9      Δco, p, q ← arg min_Δco (UpdateList);
10     R_p ← q;
       // Updates the route with least
          increase in routing metric
11     Cost ←UpdateCost(R, G);
```

for alternative high throughput and low congestion routes.

Routing algorithm. Based on the above insights, we design a "generate-and-reroute" two phase flow routing algorithm to minimize the maximal node congestion in the network. To facilitate our presentation, we define the *path congestion* of a node as the maximal congestion of all nodes in the routing path from this node to the edge server.

Our flow routing algorithm (Algorithm 1) takes the interference map I and the relay network topology $G =< C, L >$ as the input. Interference map I records the list of peer nodes that each camera node can overhear in the network. Camera nodes carry sense the list of nodes that cause interference and send periodic update to the edge server during runtime. In the relay topology G, C is the set of nodes and L is the set of possible connections.

Spider's routing algorithm first builds a shortest path tree from camera nodes to the edge server using Dijkstra's algorithm (Line 1, *generate phase*). The generated shortest path tree ensures every node in the network achieves the best pair-wise throughput to the edge server but usually leads to over-congested nodes in the network. The algorithm then alleviates the radio congestion of the bottleneck node by shifting nodes that contribute to its congestion to other routing paths (Line 2-11, *reroute phase*). In each iteration, the algorithm first re-computes path congestion (Line 3) and then selects a reroute option that leads to a minimal increase of the node ETT (Lines 4-9). The requirement for the minimal increase of the node ETT ensures the alternative path also has a good throughput. After that, the algorithm updates the routing tree topology and cost (Lines 10-11). The algorithm terminates when it reaches a pre-defined iteration threshold or there is no path change that leads to a better congestion. The variable *Threshold* in Line 2 is set to the sum of the number of feasible wireless connections from each node, which ensures exploration for every possible wireless connection. This algorithm takes $O(|L|^2)$ iterations to construct the routing tree, where $|\cdot|$ is the cardinality of the set. We experimentally demonstrate the performance of Spider's routing algorithm in §4.2.1.

Topology maintenance. The edge server collects the network topology through the control plane to make a centralized routing decision. Specifically, Spider keeps two types of topology in the data plane: *long-term* and *instant*. Spider constructs and updates the long-term topology graph by measuring mmWave link throughput between all pairs of nodes when there are no link dynamics in the deployment site (*e.g.,* when the cashierless store closes at night as detected by the cameras). The amortized overhead of updating the long-term topology is reasonably low since it is only updated on a daily or weekly basis. On the other hand, the instant topology only includes links that Spider recently used/probed. Link throughput data in instant topology is time-sensitive and is set to expire with a time-to-live threshold. When making routing decisions, the edge server combines both the instant topology and the long-term topology. For links included in the instant topology, Spider uses the link throughput from instant topology. For the links that are not in the instant topology, Spider uses throughput numbers from the long-term topology.

2.3 Video bit-rate allocation

Traditionally, video streaming algorithms adapt video bit-rate based on its local buffer size and estimated network throughput. Since the decision of these algorithms fully relies on local information, the best coordination among multiple video streams is a fair bandwidth allocation [22, 27]. In real deployment, however, some cameras desire higher bandwidth due to their greater marginal benefits to the overall video analytics accuracy. For instance, given two video streams with their bit-rate-accuracy profile in Table 1, equal bandwidth sharing of a 30 Mbps wireless link leads to an average accuracy of $(0.51 + 0.18)/2 = 0.345$. However, allocating 10 Mbps to Stream 1 and 20 Mbps to Stream 2 would achieve an overall accuracy of $(0.8 + 0.32)/2 = 0.56$.

In light of this, Spider takes into account all camera streams jointly in the design of its video bit-rate allocation algorithm (*Spider-VBA*). To do so, the edge server periodically collects video samples from camera nodes to compute bit-rate accuracy (*i.e.,* resource-accuracy) profiles (§2.3.1). Spider-VBA then formulates the problem as an optimization problem with the input of both routing information and resource accuracy profiles (§2.3.2) and generates video bit-rate configurations

Bit-rate (Mbps)	5	10	15	20	25
Accuracy (Stream 1)	0.04	0.32	0.51	0.87	1.0
Accuracy (Stream 2)	0.06	0.14	0.18	0.80	1.0

Table 1: Bit-rate-accuracy profile from two real video feeds.

for each camera using an integer programming solver (§2.3.3).

2.3.1 Resource-accuracy profiling. In order to compute the resource-accuracy profile for video bit-rate allocation, every camera node periodically sends 0.2 seconds full resolution and frame-rate videos to the edge server through the data plane. Upon receiving these *profiling videos*, the edge server applies various down-sampling combinations of their resolution and frame-rate, which we describe as *knobs* K_i for the i^{th} camera, to form various resulting *video bit-rates*, $T_i(K_i)$. Then the edge server applies the video analytic model to each profiling video for content analysis (*e.g.,* object detection). Spider calculates *analytics accuracy*, $A_i(K_i)$, by following the established practice of using results from the full resolution video as the ground truth [26]. It then generates the *resource-accuracy profile*: a set of pairs $\{(A_i(K_i), T_i(K_i))\}$.

The frequency of this resource-accuracy profiling affects the performance of Spider: the more frequent, the higher profiling accuracy Spider-VBA yields. However, frequent profiling incurs high network cost due to the transmission of video samples, resulting in reduced bandwidth for data traffic and therefore lowered analytics accuracy. Experiment in §4.4.2 shows that the optimal profiling frequency is once every 20 seconds.

2.3.2 VBA problem formulation. In order to achieve the best video bit-rate allocation, we formally formulate bit rate allocation as an optimization problem. In a mmWave data relay network $G = \langle C, L \rangle$, each camera node chooses a configuration K to produce a video feed and forwards it together with video streams from its child nodes to the edge server:

$$D_i = \underbrace{T_i(K_i)}_{\text{traffic produced by } C_i} + \underbrace{\sum_{j, C_j \in \mathbb{C}(i)} D_j}_{\text{traffic produced by the descendent}} \tag{3}$$

where D_i is the video stream to be forwarded by camera node C_i; $\mathbb{C}(i)$ is the set of children of node C_i; $T_i(K_i)$ is the bit-rate of video produced by node C_i at knob K_i, which can be calculated by resource-accuracy profiling. Our goal is to find a proper configuration K_i for each camera node C_i to maximize the overall video analytics accuracy defined below:

$$\begin{aligned} &\underset{K}{\text{maximize}} && \sum_{C_i \neq C_{ES}} A_i(K_i)/(|C|) \\ &\text{subject to} && U_{C_i} \leq 1,\ C_i \in C. \end{aligned} \tag{4}$$

where $K = \{K_1, K_2, .., K_{|C|}\}$ is the set of 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 is defined as the summation of percentage of time slots to send all three types of flow (transmitting, receiving, or deference as shown in Fig. 4) The radio utilization U_{C_i} should be less than 1 since the total percentage of available time slots is less than 100%.

$$U_{C_i} = \underbrace{\underset{(C_i, p(C_i))}{ETT} D_{C_i}}_{\text{transmitting}} + \underbrace{\sum_{C_j \in \{\mathbb{C}(i)\}} \underset{(C_j, C_i)}{ETT} D_{C_j}}_{\text{receiving}} + \underbrace{\sum_{C_k \in \{\mathbb{I}(i)\}} \underset{(C_k, p(C_k))}{ETT} D_{C_k}}_{\text{deference}} \tag{5}$$

2.3.3 Spider-VBA Solution. Finding the optimal allocation plan for all camera nodes, unfortunately, is an NP-hard optimization problem (as it can be reduced to the Knapsack problem [17]). We omit the proof for the NP-hardness due to space limitations. To solve the problem in an efficient way, we reduce it to a mixed integer linear programming problem and solve it with the standard mixed integer linear programming solver [38]. We transform it into a mixed integer linear programming (MILP) problem as follows:

$$\underset{K}{\text{maximize}} \sum_{C_i \neq C_{ES}} \sum_{j} a_{ij} \cdot k_i/(|C|). \tag{6}$$

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.

(1) **Configuration selection constraint**. Each camera node C_i may only choose one configuration at a time: $\sum_j k_{ij} = 1$. The amount of video traffic produced by a camera C_i is $t_i = \sum_j t_{ij} \times k_{ij}$, where t_{ij} is the bit-rate of camera 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.

(2) **Flow relay constraint**. For a given camera C_i, the total video traffic that needs to be transmitted by this camera should equal the sum of its own video traffic and the video traffic received from its descendants.

$$d_i = t_i + \sum_{j, R_j = i} d_j \tag{7}$$

(3) **Node utilization**. Similar to the utilization constraint in Equation 4, node utilization should not exceed 100%:

$$\underset{(C_i, p(C_i))}{ETT} d_i + \sum_{C_j \in \{\mathbb{C}(i)\}} \underset{(C_j, C_i)}{ETT} d_j + \sum_{C_k \in \{\mathbb{I}(i)\}} \underset{(C_k, p(C_k))}{ETT} d_k \leq 1 \tag{8}$$

We then adopt the standard mixed integer linear programming solver [38] to solve the problem. The MILP library can solve the optimization for hundreds of nodes within one second in our edge server.

2.4 Fault tolerance and network adaptation

Putting it all together, Spider's design at network and application layer allows it to deal with both link failures and node failures, while at the same time maintaining high network throughput for the purpose of live video analytics. For link

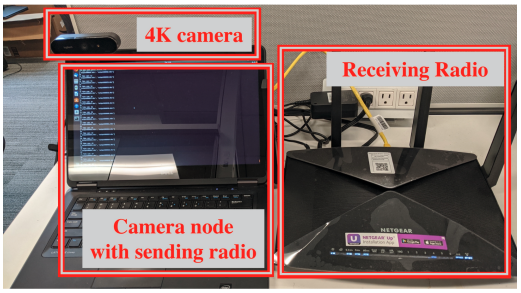


Fig. 5— Relay node hardware.

failure on the data plane, Spider detects the abrupt data rate change (*e.g.,*-30%) and reports it to the edge server through control plane immediately. The edge server then re-runs the routing and video bit-rate allocation algorithms and pushes new receiver ID and video bit-rate configurations (based on bandwidth allocation results) to each camera node through the control plane. Once the camera node switches to another link, it periodically checks the availability of the previous (blocked) link using bounded exponential backoff [10] and switches back when the blocked link recovers.

Control plane link failure recovery follows a similar procedure. Note that they are much less common in Spider compared with data plane link failure as Wi-Fi links are less susceptible to blockage. Once a camera node is unreachable in control plane, the edge server cuts off 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 request from the failure node. Once the failed node returns, the waiting thread will update its status in the routing table, and the edge server updates the routes and camera configurations accordingly.

3 Implementation

We have implemented experimental prototypes of both the Spider camera node and edge server, which we use in the cashierless store scenario below in §4.

Camera node. Each camera node is a Dell E7440 laptop [15] equipped with a Logitech BRIO 4K camera [35] and two WiGig radios:[3] a QCA9008 mmWave network interface card [5] for transmitting, and a NETGEAR Nighthawk X10 router [39] (802.11ac&ad dual band) for receiving. To reduce the association delay, 802.11ad band of all the routers is configured into roaming mode that shares the same BSSID. Each node uses the default beam selection mechanism in 802.11ad [23] in the physical layer, and standard CSMA/CA with RTS/CTS in MAC layer. In the control plane, all camera nodes connect to the edge server router (AP mode) with 802.11ac.

To increase the throughput of 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 instead of using a software router [31]. Such a design offloads the majority of computation for packet header processing from CPU to network interface card, and therefore allows camera node to forward hundreds of concurrent video streams by processing incoming video traffic at 2 Gbps using a single CPU core. The camera nodes encode their video stream into two-second length segments with H.264 codec. To reduce network traffic, each camera also invokes a background subtraction (BGS) [8] module as an early filter to only send frames with moving objects. Video forwarding logic and BGS module are implemented in C++ to ensure execution efficiency.

Edge server. The edge server is customized from Lambdalabs [32] equipped with a 10-core Intel Core i9 CPU, and two NVIDIA RTX 2080 Ti GPUs. Aside from Spider routing and Spider-VBA algorithms, it runs cascaded analytics pipelines [34, 37] for efficient video processing. As a result, Spider is able to process HD videos on one edge server at 370 fps, 45× faster than the baseline of running DNNs on all video frames.

4 Evaluation

We present our experimental evaluation of Spider in this section, with the following highlights:

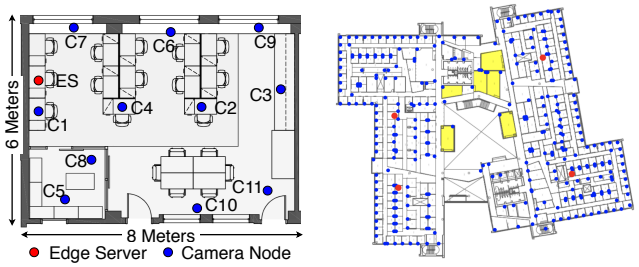
(1) Given a target video analytic accuracy, Spider improves the maximum-supported camera density by up to 2.76× compared with best-performing comparison scheme.

(2) In the presence of network failures, Spider's dedicated control plane reduces failure recovery latency and negative impact on video analytic accuracy by 5.4-100× and 32% over baselines.

4.1 Experimental Methodology

Use Cases. We evaluate Spider in two use cases: cashierless store and security monitoring. For the **cashierless store** application, we use the testbed deployment shown in Fig. 6(a). According to our estimation, a typical cashierless store (*e.g.* Amazon Go [2]) has 0.25 cameras per square foot. To match the density requirement of a real cashierless store, we configure each physical camera nodes to send multiple virtual camera streams so that the total number of streams sent in our deployment testbed matches the expected number of streams sent in a real cashierless store application. For **security monitoring**: we run trace-driven simulation based on the floor plan of a 68,880 square feet enterprise building (Fig. 6(b)). The security monitoring application needs a larger coverage but a lower camera density compared with the cashierless store. If we deploy cameras to cover every room of the floor shown in Fig. 6(b), the camera density would be 4.5 cameras per 1,000

[3]Technically, one WiGig radio is needed per camera. However, as the 802.11ad Linux driver, wil6210 [58], does not support peer to peer mode as yet, we adopt two WiGig radios as a proof-of-concept design.



(a) Deployment of an 11-camera testbed in an 830 square feet room to emulate a **cashierless store** scenario.

(b) A simulated 68,880 square feet office building, to emulate a **security monitoring** scenario.

Fig. 6— Floor plans of two experimental scenarios.

square feet. To run the trace based simulation, we calculate link throughput based on the floor plan and mmWave signal propagation models [59, 66]. In addition, we incorporate link blockages and failures into the simulation based on the percentage of failure in the testbed.

Routing Baselines. Many works focus on mesh routing, but few focus on mesh for mmWave,[4] so we pick the most representative routing protocol for each of two different categories of routing algorithm: max-throughput routing and shortest-path routing:

- **Roofnet** [11] is a pioneering mesh network that aims to maximize throughput and is used in products from Meraki and Cisco.
- **Batman** [28] is a shortest path routing protocol implemented in the official Linux kernel.

In addition to aforementioned routing protocols on mmWave network, we also compare Spider with a baseline that streams video with **802.11ac**. In this baseline, the camera nodes directly connect to the edge server with single hop 802.11ac Wi-Fi connections.

Video Bitrate Allocation Baseline. We use a fair video bitrate allocation (**FVBA**) as the baseline. FVBA is equivalent to a centralized implementation of FESTIVE [27], which allocates the fair bandwidth to each camera nodes through control plane rather than letting each camera node run bitrate adaptation distributedly.

Video analytics workloads. We evaluate Spider with three representative video analytic workloads: **object detection** [46], **face detection** [62], and **scene text detection** [65]. These three workloads show different sensitivities to video knobs. We follow the convention of the prior literature and use fully-fledged neural networks as the ground truth to label videos (at the highest resolution and frame-rate) [26]. We adopt the *F1 score*, a standard accuracy metric taking into account both precision and recall as the primary metric in our evaluation [26].

[4]We are aware of Facebook Terragraph as a mmWave based mesh network, but we are not able to find its routing algorithm details in the existing literature for comparison purpose.

Following convention in the Computer Vision community, we consider an output bounding box as a true positive detection only if the Intersection over Union (IoU) between the box and the ground truth is above 0.5 [46].

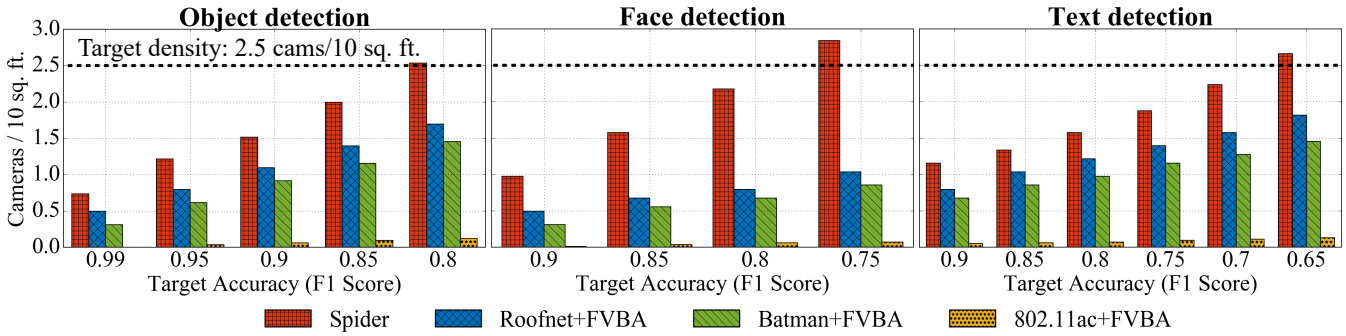
4.2 Accuracy versus Density

We conduct a field study to answer the question: given an accuracy requirement, what is the deployment density that Spider can support?

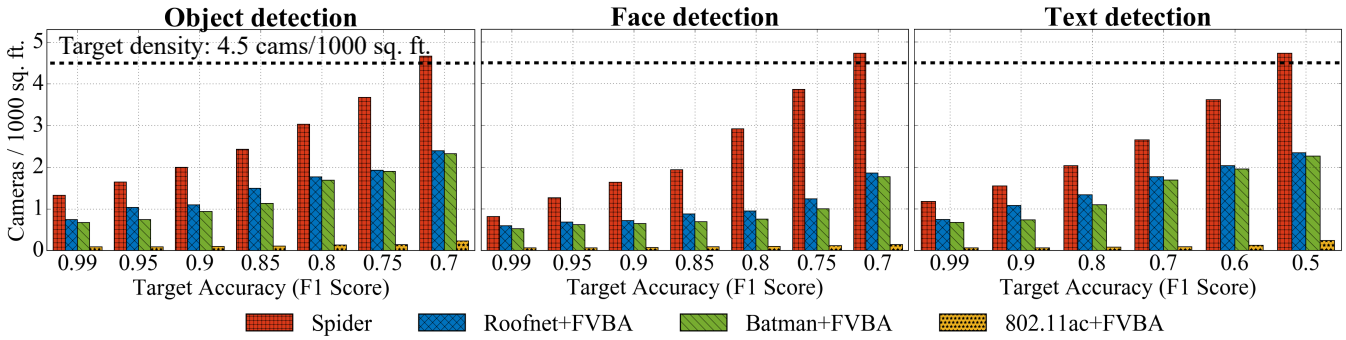
For a wireless video camera network, a denser camera deployment reduces the average video accuracy since each camera share less wireless bandwidth to transmit its video. We compare the maximum camera density by Spider and baselines for a given accuracy requirement. For cashierless store application, the density is the number of virtual camera streams per 10 square feet. The security monitoring use case has a larger coverage but lower density requirement. We measure its density by the number of virtual camera streams per 1000 square feet.

Fig. 7(a) compares the performance of Spider with three baselines on three workloads for the cashierless store use case. The baseline Roofnet+FVBA and Batman+FVBA are mmWave camera networks that use Roofnet and Batman as the routing protocols, respectively. The baseline 801.11ac+FVBA builds the video network upon 802.11ac Wi-Fi. All baselines use FVBA as the bitrate allocation mechanism. From the figure, we observe that, as we release the accuracy requirement, the maximum camera density that a camera network can support increases. Spider reaches the target camera density for a cashierless store (2.5 cameras per 10 square feet) under the accuracy requirement of 0.8, 0.75, 0.65 for object detection, face detection, and text detection workloads, respectively. Roofnet+FVBA achieves the density of 1.7, 1.03, and 1.82 cameras per 10 square feet for three workloads and Batman+FVBA achieves the density of 1.46, 0.85, and 1.46 cameras per 10 square feet for three workloads under the corresponding accuracy requirement. Spider improves the camera deployment density by 1.46–2.76× compared with Roofnet+FVBA and 1.74–3.34× compared with Batman+FVBA. The improvement is even higher when comparing with 801.11ac+FVBA.

We also compare the performance of Spider with three baselines for the security monitoring use case. The target camera density is 4.5 cameras per 1000 square feet. As Fig. 8 shows, Spider reaches the target density at the accuracy requirement of 0.7, 0.7, and 0.5 for three workloads. Correspondingly, Roofnet+FVBA achieves the density of 2.4, 1.86, and 2.34 cameras per 1000 square feet for three workloads and Batman+FVBA achieves the density of 2.32, 1.76, and 2.26 cameras per 1000 square feet for three workloads. Spider improves the camera deployment density by 1.95–2.54× compared with Roofnet+FVBA and 2.01–2.69× compared with Batman+FVBA.



(a) Cashierless store use case, targeting 2.5 cameras/10 sq. ft.



(b) Security monitoring use case, targeting 4.5 cameras/1,000 sq. ft.

Fig. 7— *Achievable density*: Camera density supported by Spider and baselines given a specific accuracy requirement for two use cases, on object detection, face detection, and text detection workloads.

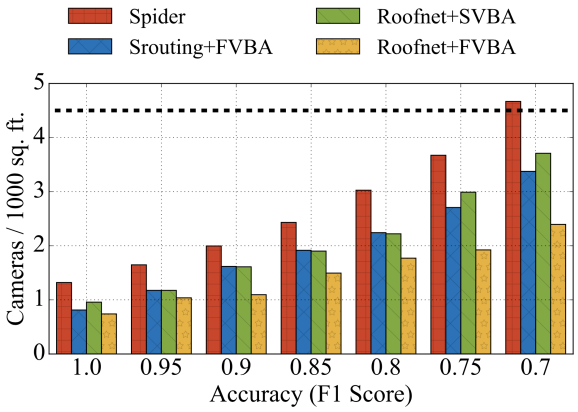


Fig. 8— *Effect size analysis:* Contribution of each Spider design component on maximum supported camera density on object detection, compared to Roofnet+FVBA baseline.

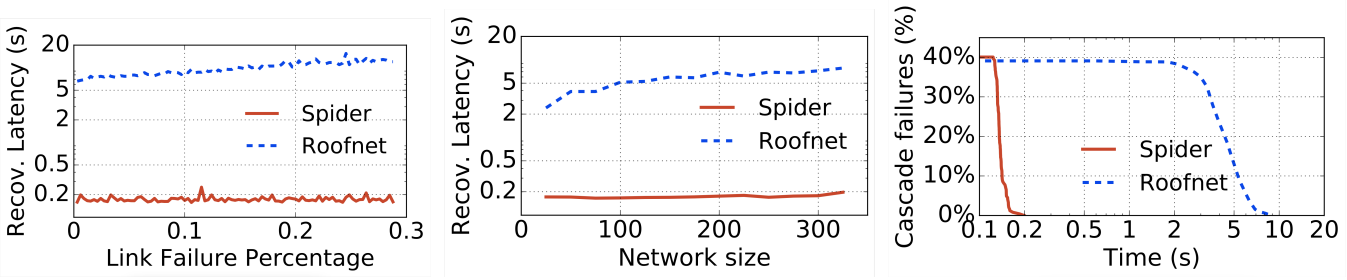
In summary, Spider improves maximum camera deployment density by up to 2.76× compared with Roofnet+FVBA and up to 3.34× compared with Batman+FVBA.

4.2.1 Ablation study. We conduct an ablation study to understand the effectiveness of each design component in Spider. We use Roofnet+FVBA as the baseline system in the study since it has the best performance among all baselines. To evaluate the effectiveness of Spider's routing algorithm, we combine Spider's routing (Srouting) with FVBA used in the baseline. Similarly, we combine Spider's video bitrate allocation algorithm (SVBA) with Roofnet to evaluate the contribution of Spider's VBA algorithm. Fig. 8 shows the performance of Srouting+FVBA and Roofnet+SVBA. We observe that Spider's bit rate allocation algorithm has a slightly higher contribution than Spider's routing algorithm. Performance gain brought by these two modules is as high as 55% and 45%, respectively.

4.3 Robustness of Spider

We evaluate failures from two aspects: *i)* how fast Spider can recover from a failure? *ii)* what is the impact of failure on Spider's accuracy? Similarly, we adopt Roofnet as the default algorithm for comparison.

4.3.1 Failure recovery latency. When link failure happens, the affected camera node has to search for a new node for data forwarding. The latency, especially message propagation latency, for this re-routing is critical to network scalability. Table 2 shows average failure recovery latency in the testbed. Spider is by 100× faster in message propagation and 5.4× faster in total re-routing process (81-99% reduction in message propagation latency and total re-routing latency respectively). To further demonstrate the scalability of Spider, we



(a) Latency of failure recovery versus link failure ratio.

(b) Latency of recovery versus the size of camera network under 10% link failure.

(c) The process of failure recovery under 10% link failure.

Fig. 9— Failure recovery latency.

Algorithm	Neighbor disc.	Msg. prop.	Re-asso.	Total
Spider	N/A	**6ms**	0.15s	**0.15s**
Roofnet	80ms	**0.58s**	0.15s	**0.81s**

Table 2: Time to recover from a single link failure in the cashierless store testbed.

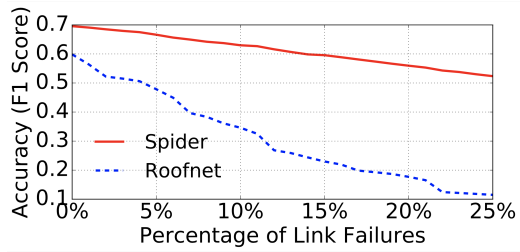


Fig. 10— Video analytic accuracy under different link failure percentages.

run simulation experiments on the floor plan shown in Fig. 6. To understand the reliability of Spider compared with our baseline, we intentionally include experiments with a wide range of failure rates, from no failure to an extreme link failure rate (30%). We define the failure recovery latency as the duration during which 95% percent of nodes recover from the network failure. We then examine the failure recovery latency of Spider and Roofnet in various settings. Specifically, we focus on three aspects of failure recovery:

The impact of different percentage of link failure on re-routing latency. We observe the re-routing latency on Roofnet grows from 6.6 to 12 seconds with an increasing number of failed links. In contrast, Spider achieves a relatively constant re-routing latency (less than 0.2 seconds) as the number of failure links grows from 1% to 30% (Fig. 9(a).

The impact of different network sizes on re-routing latency. We observe the re-routing latency grows gradually on Roofnet as the network size grows: from 2.4 to 7.8 seconds as the number of camera nodes grows from 25 to 330. In contrast, the re-routing latency on Spider maintains less than 0.2 seconds as the network size grows (Fig. 9(b)). This result demonstrates that Spider is more scalable than Roofnet in the presence of network failure.

The process of failure recovering. We observe that 10% of link failures can lead to extra 40% of cascade failures to both systems due to the relay architecture. However, Spider only takes 0.2 seconds to recover from the failure. In contrast, due to the overhead of broadcasting in the multi-hop data plane, Roofnet starts recovery after 2 seconds and finishes at 7.8 seconds, taking 39× the time Spider uses (Fig. 9(c)).

4.3.2 Impact on accuracy.

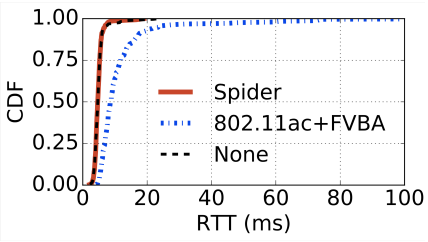
We use our simulation setup for security monitoring, which it is easy for us to obtain a failure-free baseline for comparison, to evaluate the impact of failure on video analytics performance. When a link fails, cameras that forward its stream through that link cannot send its video streams, leading to degradation for average accuracy for the whole video analytic network. We randomly shut down different percentage of links and compute the average accuracy of video streams of the overall video analytic network. For comparison, we also run Roofnet in the same network topology settings. We show the impact of link failures in Fig. 10. As expected, the average accuracy decreases with the growing number of link failures. However, accuracy of Spider drops much slower than Roofnet. This result demonstrates that failures lead to much smaller performance degradation in Spider compared with Roofnet.

4.4 Sensitivity analysis

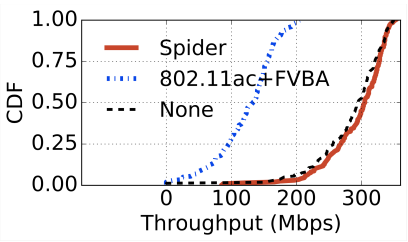
We conduct the sensitivity analysis to Spider from the following perspectives.

4.4.1 Interference to legacy Wi-Fi users.

We quantify the impact of Spider control packets on legacy Wi-Fi users in these experiments by comparing Spider with 802.11ac+FVBA baseline. We put a legacy Wi-Fi device (a laptop) in our cashierless store testbed and configure it to work on the same channel as Spider's control plane. This laptop ping a Google server 1,000 times. We then measure the round trip time (RTT) with and without launching Spider. Fig. 11(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 802.11ac+FVBA



(a) Network latency.



(b) Network throughput.

Fig. 11— Impact of Spider's control plane on legacy Wi-Fi users.

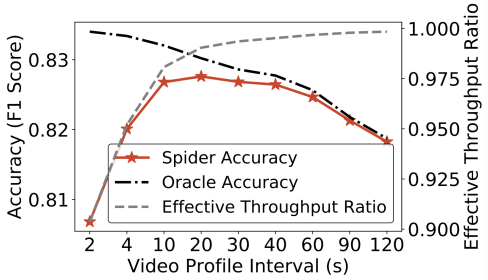


Fig. 12— Impact of video profiling interval on accuracy.

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 802.11ac+FVBA transmits video streams over the same Wi-Fi band. Running 802.11ac+FVBA thus introduces 16.2 ms extra delay, 81× higher than the latency 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. Fig. 11(b) shows the CDF of network throughput. We observe that this legacy Wi-Fi device can achieve almost the same network throughput regardless of the presence of Spider. In contrast, 95% percentile of the peak throughput drops significantly from 305.0 Mbps to 133.5 Mbps when 802.11ac+FVBA is running.

4.4.2 Impact of video profiling. Each camera node in Spider periodically transmits profiling video segments to the edge server over 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 Figure 12. Spider's average F1 score grows rapidly as we increase the video profiling interval from 2s to 20s. 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 F1 score begins to drop at 20s interval settings. Spider achieves 82.7% average F1 score in 20s interval settings, 0.5% lower than the oracle. At the same time, the effective throughput ratio maintains at a high level (>99%). Suggested by this result, we choose 20s as the default video profiling interval in our design.

Network size	50	100	200	300
Spider-Routing Time(s)	0.011	0.025	0.150	0.334
Spider-VBA Time(s)	0.578	0.590	0.629	0.680

Table 3: Time cost for Spider's routing and VBA algorithm.

4.4.3 Computational cost. We present the computational cost for routing and video bitrate allocation algorithm of Spider. Table 3 shows the time cost of Spider-Routing and Spider-VBA on the edge server. For a 300-node network, the total time cost for two algorithm is around one second, whereas the length of video chunk streamed in Spider is two seconds. That means, Spider is capable to compute the optimal configuration according to the network change for every chunk of video streamed in its network.

5 Discussion and future work

Control plane coverage. Spider needs to ensure the control plan coverage is not smaller than the data plane coverage. While the Spider prototype uses single hop Wi-Fi as the control plane, the communication range of Wi-Fi does not fundamentally limit its scalability because, as the deployment area scales up, Spider can use as its control plane an enterprise Wi-Fi or wide-area cellular network. We leave the implementation and evaluation of these alternate control planes as future work, but note that these designs require minimal changes, well understood to yield both low latency and high reliability control plane performance.

Topology maintenance overhead. While Spider's topology maintenance overhead (§2.2) may grow with greater mmWave link dynamics from moving objects or people, Spider only reacts to link failures or abrupt drops in throughput for established mmWave connections, which, according to §4.3, take *ca.* 0.15 s using our agile rerouting design.

Interference on mmWave band. Different from Wi-Fi, the mmWave link communicates through highly directional beam. The mmWave link will not interfere with other devices that

are not in its beamforming direction. Therefore, interference from other devices has a much less impact on mmWave links. We leave the medium sharing design between Spider and other mmWave links into future work.

Computational load. There are two types of computational loads when operating Spider system. One is the scheduling load, which is to compute the optimal route and bitrate configuration for each camera node. The other is video analytic load. In our implementation, we run both computational loads on a dedicated edge server (One CPU core for the scheduling load and nine CPU cores plus two GPUs for the video analytic load). Given a less powerful edge server or more camera nodes, we expect can still handle the scheduling load with the computational resources on the edge server. The video analytic load could be partially offloaded to the cloud. We leave the offload between edge and cloud as the future work.

6 Related Work

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

Wireless video analytics systems. Live video analytics are becoming more pervasive due to the increasing number of cameras and the deployment of edge computing devices [21, 25, 26, 30, 60, 61, 63]. Among those works, wireless video analytics systems like Vigil [63] and ZC^2[60] demonstrate flexibility in their deployment [60, 63]. In contrast, Spider leverages a much higher throughput mmWave mesh, as opposed to the slower Wi-Fi or harder to deploy wired links adopted by these systems.

mmWave networks. mmWave communication is emerging as one of the key technologies in the 5G era. However, the short communication range and high fragility of mmWave links pose challenges in fully utilizing their bandwidth. Efforts have been spent in the research community to model mmWave networks [66], improve mmWave link coverage [54, 56, 57], reduce beam alignment and AP switching delay [20, 43, 64], and build multi-radio hybrid networks with Wi-Fi or LTE [52]. mmWave has also been used to stream videos that demand high bandwidth and low latency [9]. However, most of these works focus on single hop mmWave links, while Spider builds a relay network on top of mmWave and tackles communication range and link dynamics issues to build a useful video analytics system.

In addition, fast beam alignment algorithms [20, 43, 64] cannot solve the problem of information broadcasting delay in Spider. When a node A wants to connect to another node B in the mmWave relay network. Node B might be busy with its ongoing transmission, therefore node A has to wait unit the ongoing transmission on node B finishes, which inevitably introduces extra delays.

Wireless mesh networks. Multi-hop relay technologies have been extensively studied in wireless mesh networks [11, 12, 16, 33, 55]. In the literature, there are two types of clustered routing algorithms in relay networks: pro-active routing algorithms [14, 45] and on-demand routing algorithms [29, 44]. However, both types of routing algorithms do not fit in Spider, due to the overhead of message broadcasting in mmWave mesh networks. Spider solves the problem by introducing a dedicated long-range control plane to centralized decision-making for routing.

There is also a relay network design based on 60 GHz radios: Terragraph [55]. Unlike Terragraph's focus on network throughput optimization, Spider focuses on application-driven metrics, maximizing video analytics accuracy through a joint optimization of network flow and application configuration. In addition, Terragraph targets deployments with more stable link dynamics, *e.g.* cell towers, roofs, which are well above the ground and thus less likely to experience blockage. Therefore Terragraph can transmit its control packets through mmWave links without harming overall performance. In contrast, Spider may be deployed for applications with frequent mmWave link blockages (cashierless stores, where camera nodes might be deployed in locations that can be cut off as customers walk by). To solve the blockage issue, Spider takes a different approach in network architecture design, which separates low-frequency control plane for network control.

Software-Defined Wireless Networking. Decoupling control plane from data plane has been explored in the wireless networks as well [13, 49, 50]. However, all these works assume the control plane residences at the same frequency band as data plane, and thus shares the same physical propagation characteristic. In contrast, the control plane and data plane in Spider are decoupled over frequency. This allows Spider to transmit control packets over robust wireless links at a lower frequency, and data packets over high-throughput links at a higher frequency.

7 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.

8 Acknowledgement

We thank the reviewers and our shepherd, Yifan Zhang for their insightful comments. This research was supported by a gift from the Microsoft Corporation.

References

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

[2] Amazon is expanding its cashierless Go model into a full-blown grocery store. Webpage.

[3] Amazon Go. Webpage.

[4] H. Assasa, S. K. Saha, A. Loch, D. Koutsonikolas, J. Widmer. Medium access and transport protocol aspects in practical 802.11ad networks. *IEEE WoWMoM*, 2018.

[5] Atheros QCA9008 TBD1 Wireless AC+AD Dual Band Wi-Fi Card. Webpage.

[6] AWS Outposts. Webpage.

[7] Azure Stack Edge. Webpage.

[8] Background Subtraction Methods - OpenCV. Webpage.

[9] G. Baig, J. He, M. A. Qureshi, L. Qiu, G. Chen, P. Chen, Y. Hu. Jigsaw: Robust Live 4K Video Streaming. *ACM MobiCom*, 2019.

[10] V. Bharghavan, A. Demers, S. Shenker, L. Zhang. MACAW: a media access protocol for wireless LAN's. *ACM SIGCOMM*, 1994.

[11] J. Bicket, D. Aguayo, S. Biswas, R. Morris. Architecture and evaluation of an unplanned 802.11b mesh network. *MobiCom*, 2005.

[12] S. Biswas, R. Morris. Opportunistic Routing in Multi-Hop Wireless Networks. *ACM SIGCOMM Compute Communication Review*, 2004.

[13] A. Cidon, K. Nagaraj, S. Katti, P. Viswanath. Flashback: Decoupled lightweight wireless control. *ACM SIGCOMM Computer Communication Review*, 2012.

[14] T. Clausen, P. Jacquet. RFC3626: Optimized link state routing protocol (OLSR), 2003.

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

[16] R. Draves, J. Padhye, B. Zill. Routing in multi-radio, multi-hop wireless mesh networks. *ACM MobiCom*, 2004.

[17] A. Fréville. The multidimensional 0–1 knapsack problem: An overview. *European Journal of Operational Research*, 2004.

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

[19] 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.

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

[21] 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.

[22] T.-Y. Huang, R. Johari, N. McKeown, M. Trunnell, M. Watson. A buffer-based approach to rate adaptation: Evidence from a large video streaming service. *ACM SIGCOMM*, 2014.

[23] IEEE Standards 802.11ad-2012: Enhancements for Very High Throughput in the 60 GHz Band. *IEEE Standard Association*, 2012.

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

[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] J. Jiang, V. Sekar, H. Zhang. Improving fairness, efficiency, and stability in http-based adaptive video streaming with festive. *CoNEXT*, 2012.

[28] D. Johnson, N. Ntlatlapa, C. Aichele. Simple pragmatic approach to mesh routing using BATMAN, 2008.

[29] D. B. Johnson, D. A. Maltz. Dynamic source routing in ad hoc wireless networks. *Mobile computing*. Springer, 1996.

[30] 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.

[31] E. Kohler, R. Morris, B. Chen, J. Jannotti, M. F. Kaashoek. The Click modular router. *ACM Transactions on Computer Systems (TOCS)*, 2000.

[32] Lambda Labs. Webpage.

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

[34] Live Video Analytics with Microsoft Rocket for reducing edge compute costs. Webpage.

[35] Logitech BRIO Webcam with 4K Ultra HD video. Webpage.

[36] M. H. Mazaheri, S. Ameli, A. Abedi, O. Abari. A millimeter wave network for billions of things. *ACM SIGCOMM*, 2019.

[37] Microsoft Rocket Video Analytics Platform. Webpage.

[38] MILP Solver. Webpage.

[39] NETGEAR Nighthawk X10 router. NETGEAR.

[40] T. Nitsche, C. Cordeiro, A. B. Flores, E. W. Knightly, E. Perahia, J. C. Widmer. IEEE 802.11ad: directional 60 GHz communication for multi-Gigabit-per-second Wi-Fi. *IEEE Communications Magazine*, 2014.

[41] NY Daily News: 70% of NYC subway stations now have security cameras. Webpage.

[42] Oregon City Schools expanding security camera system. Webpage.

[43] J. Palacios, D. Steinmetzer, A. Loch, M. Hollick, J. Widmer. Adaptive Codebook Optimization for Beam Training on Off-the-Shelf IEEE 802.11ad Devices. *ACM MobiCom*, 2018.

[44] C. Perkins, E. Belding-Royer, S. Das. RFC3561: Ad hoc on-demand distance vector (AODV) routing.

[45] C. E. Perkins, P. Bhagwat. Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers. *ACM SIGCOMM*, 1994.

[46] J. Redmon, S. Divvala, R. Girshick, A. Farhadi. You only look once: Unified, real-time object detection. *IEEE CVPR*, 2016.

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

[48] A. Schrijver. On the history of the transportation and maximum flow problems. *Mathematical Programming*, 2002.

[49] J. Schulz-Zander, C. Mayer, B. Ciobotaru, S. Schmid, A. Feldmann. OpenSDWN: Programmatic control over home and enterprise WiFi. *ACM SOSR*, 2015.

[50] J. Schulz-Zander, L. Suresh, N. Sarrar, A. Feldmann, T. Hühn, R. Merz. Programmatic orchestration of Wi-Fi networks. *USENIX ATC*, 347–358, 2014.

[51] 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*, 2019.

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

[53] S. Sur, V. Venkateswaran, X. Zhang, P. Ramanathan. 60 GHz indoor networking through flexible beams: A link-level profiling. *ACM SIGMETRICS*, 2015.

[54] S. Sur, X. Zhang, P. Ramanathan, R. Chandra. BeamSpy: enabling robust 60 GHz links under blockage. *NSDI*, 2016.

[55] Terragraph: Solving the Urban Bandwidth Challenge. Webpage.

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

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

[58] Wil6210. Webpage.

[59] 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*, 1997.

[60] 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.

[61] 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.

[62] K. Zhang, Z. Zhang, Z. Li, Y. Qiao. Joint face detection and alignment using multitask cascaded convolutional networks. *IEEE Signal Processing Letters*, 2016.

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

[64] 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.

[65] X. Zhou, C. Yao, H. Wen, Y. Wang, S. Zhou, W. He, J. Liang. EAST: an efficient and accurate scene text detector. *IEEE CVPR*, 2017.

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