Poster Abstract: Reducing Tail Response Time of Vehicular Applications

HyunJong Lee, Jason Flinn
University of Michigan
{hyunjong, jflinn}@umich.edu

Today's vehicles are sensor-rich but computation-poor. To better assist drivers, current vehicles have a large number of diverse sensors; for instance, the 2017 Ford GT has over 50 built-in cameras and sensors [2] that can determine speed, location, humidity, occupancy, mechanical positioning, and a wealth of other data. However, modern vehicles have little general-purpose computing capacity due to cost, maintenance, and survivability concerns. For instance, vehicle manufacturers aim for vehicles to last for 20 years, and any general-purpose computing platforms would become obsolete or need maintenance many times during that lifespan.

Emerging vehicular applications often require substantial computation to process rich sensor data; for example, a parking spot locator may need to process video from an external camera. Since the needed computation is not available in the vehicle, such applications offload the processing of sensor data to other platforms. Two current approaches are offloading computation to the cloud (e.g., OnStar's ATOMS [5]) and offloading computation to edge mobile devices such as smartphones within the vehicle (e.g., AppLink [1]). In the future, edge computing platforms located in cellular infrastructure or at roadside hotspots may provide additional locations for hosting computation. Figure 1 shows these three possibilities.

The challenge in offloading computation over vehicular sensor data is that response times can vary substantially due to mobile network quality and unpredictable load changes on the platforms hosting offloaded computation. Sensor-rich vehicular applications are usually user-facing, so providing low response time is vital for a good consumer experience and minimizing driver distraction. However, low average response time is not enough; response times must also be consistent (i.e., the tail response time should also be low) for a good user experience [3].

Our proposed solution is to use passive measurement and historical data to estimate network latency and compute times for offloaded sensor processing in vehicular applications. Based on these measurements, we will select the cloud, roadside, or mobile phone platform that yields the fastest predicted response time.

However, network predictions are inherently uncertain in vehicular environments due to high rates of mobility. In addition, edge devices on the road and in the vehicle can have unpredictable load spikes due to limited compute capacity and competing demand from other applications and nearby vehicles (in the case of roadside devices). Therefore, we propose to selectively employ *redundancy* to reduce the response time tail. When response time predictions have a high degree of uncertainty, we will replicate offloaded computation on multiple platforms and use the fastest response. In addition, there may be multiple mobile network paths for data transmission; e.g., we can use WiFi or cellular networks to reach the cloud, and an edge device may use multiple networks to retrieve data from cloud storage. Normally, we will send requests and responses over the fastest network; however, when estimates of network latency are uncertain, we will also replicate data transmission by sending requests over multiple networks to reduce tail response time.

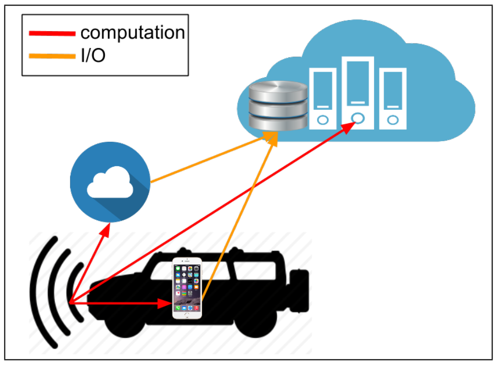

Fig. 1: Sensor data processing is offloaded to mobile, roadside, and cloud platforms. Offloaded computation may need additional data from cloud storage.

Selecting locations for sensor-data processing: A primary goal for offloading vehicular sensor-data processing is minimizing the application response time. As with prior offloading systems [4], we predict application latency by first estimating the supply and demand of network and computational resources and then calculating the estimated time for each potential offload site to produce a response. We can then run the computation on the site with the lowest predicted response time.

However, prior prediction mechanisms for offload have heavily relied on measurements of network [10] and computation load [4]. This approach works well if measurements have been taken recently and the measured resources are relatively stable. However, in vehicular scenarios, sensor-processing may be relatively infrequent, so past measurements may be too stale. Also, high rates of mobility may lead to rapid changes in network quality or the set of edge devices located near the vehicle.

Therefore, we are using a hybrid approach that combines passive measurements with crowd-sourced historical data collected over a long time period. When measurements are stale, the hybrid approach will rely more on crowd-sourced historical data; when measurements are recent, they will be the primary contributor to the predictions. The hybrid approach also affects the confidence in the predictions; recent measurements are likely to be more accurate than historical predictions.

Offload redundancy: Given that vehicular scenarios are likely to exhibit much more variability in environmental conditions than scenarios with little or no mobility, we expect that the offload site selections based on estimates of those conditions will be incorrect more often. Missed predictions mean that the latency experienced by the user may be quite high since the computation is running at the wrong site. This can be a substantial contributor to tail latency.

We plan to selectively employ redundancy in offloading to reduce tail latency [8]. Rather than employing redundancy at all times as is done in system like Tango [6], we will consider the *uncertainty* in our underlying network and computation load predictions. If we are relatively confident in those predictions (e.g., because they are based on recent measurements), we will simply select the best predicted site to offload computation. If we are not confident in those predictions (e.g., because they are based on historical observations with high variance), we may choose to run the offloaded computation on multiple sites and use results from the fastest site to respond.

Figure 2 shows the potential benefit of this approach. The rightmost two lines shows the CDF of the latency distribution for two potential offloading sites. Choosing the best site moves the expected response time from the rightmost line to the middle one. However, redundant execution improves the expected response time even further, as shown by the leftmost line in the graph. Importantly, the 99% response time improves by 200ms.

However, redundancy comes at a cost: for example, extra cellular data usage or mobile device energy consumption. Thus, redundancy should only be employed when the expected benefits outweigh those costs.

Network redundancy: Network redundancy can also improve tail response times. The vehicle may have diverse connectivity options (e.g., Wifi, cellular, and Bluetooth) for communicating with cloud and edge devices, and

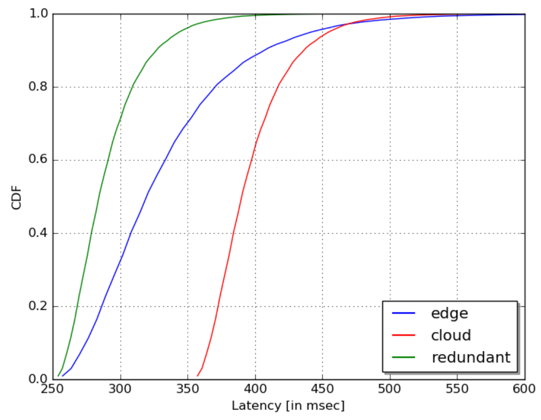

Fig. 2: Latency CDF for three potential offload decisions

those devices may have similar diverse options for communicating with cloud data sources. When multiple networks are available, sending data over multiple paths can reduce overall service response time.

Multipath TCP (*MPTCP*), in a recently proposed and standardized variant of TCP [7, 11] that stripes data over multiple *subflows*, which are connections over different paths between two endpoints. We are modifying MPTCP to selectively employ redundant transmission over such subflows with the goal of reducing service response time. By calculating the uncertainty in each subflow latency measurement, we determine the predicted benefit of redundant transmission. When that benefit exceeds the cost of using additional network resources, we send data over multiple subflows and the receiver discards any redundant data that it receives.

Our initial design supports unmodified applications by inferring that small transmissions are likely to be latency-sensitive and will benefit from redundancy. Larger transmissions MPTCP striping as normal. Applications that are modified to provide intentions [9] about which data is latency-sensitive can receive more targeted benefits.

References Cited

[1] AppLink. https://developer.ford.com/pages/applink.
[2] Ford GT Electronic Sensors. https://goo.gl/eXuwB7.
[3] J. Dean and L. A. Barroso. The tail at scale. *Communications of the ACM*, 56(2):74–80, 2013.
[4] J. Flinn. *Cyber Foraging: Bridging Mobile and Cloud Computing*. Morgan and Claypool Publishers, 2012.
[5] S. Gallagher. Onstar gives volt owners what they want: their data, in the cloud. 2012.
[6] M. S. Gordon, D. K. Hong, P. M. Chen, J. Flinn, S. Mahlke, and Z. M. Mao. Accelerating mobile applications through flip-flop replication. In *MobiSys*. ACM, 2015.
[7] M. Handley. TCP Extensions for Multipath Operation with Multiple Addresses. RFC 6824, Jan. 2013.
[8] B. D. Higgins, K. Lee, J. Flinn, T. J. Giuli, B. Noble, and C. Peplin. The future is cloudy: Reflecting prediction error in mobile applications. In *MobiCASE*. IEEE, 2014.
[9] B. D. Higgins, A. Reda, T. Alperovich, J. Flinn, T. J. Giuli, B. Noble, and D. Watson. Intentional networking: opportunistic exploitation of mobile network diversity. In *MobiCom*. ACM, 2010.
[10] V. Jacobson. Congestion avoidance and control. In *SIGCOMM*. ACM, 1988.
[11] D. Wischik, C. Raiciu, A. Greenhalgh, and M. Handley. Design, implementation and evaluation of congestion control for multipath tcp. In *NSDI*, 2011.