A Robust Load Balancing and Routing Protocol for Intra-Car Hybrid Wired/Wireless Networks

Wei Si, David Starobinski, and Moshe Laifenfeld

Abstract—With the emergence of connected and autonomous vehicles, sensors are increasingly deployed within cars to support new functionalities. Traffic generated by these sensors congest traditional intra-car networks, such as CAN buses. Furthermore, the large amount of wires needed to connect sensors makes it harder to design cars in a modular way. To alleviate these limitations, we propose, simulate, and implement a *hybrid* wired/wireless architecture, in which each node is connected to either a wired interface or a wireless interface or both. Specifically, we propose a new protocol, called *Hybrid-Backpressure Collection Protocol (Hybrid-BCP)*, to efficiently collect data from sensors in intra-car networks. Hybrid-BCP is backward-compatible with the CAN bus technology, and builds on the BCP protocol, designed for wireless sensor networks. We theoretically prove that an idealized version of Hybrid-BCP achieves optimal throughput. Our testbed implementation, based on CAN and ZigBee transceivers, demonstrates the load balancing and routing functionalities of Hybrid-BCP and its resilience to DoS attacks and wireless jamming attacks. We further provide simulation results, obtained with the ns-3 simulator and based on real intra-car RSSI traces, that compare between the performance of Hybrid-BCP and a tree-based data collection protocol. Notably, the simulations show that Hybrid-BCP outperforms the tree-based protocol on throughput by 12%. The results also show that Hybrid-BCP maintains high packet delivery rate and low packet delay for safety-critical sensors that are directly connected to the sink through wire.

Index Terms—Intra-Vehicular, Hybrid Networks, Load Balancing, Routing Protocol.

1 Introduction

The conventional intra-car communication model, in which sensors communicate with Electronic Control Units (ECUs) via Controller Area Network (CAN) buses, faces several limitations [1]. CAN buses are indeed stressed to their limit due to constant increase in electrical contents and sensors. As electrical architectures often serve over multiple vehicle platforms and multiple model years, newly added electrical components may increase traffic load over the network beyond its originally designed capacity requiring costly and undesired network wired extensions. Thus, intra-car wired network architectures come with attendant costs of weight, traffic congestion, and maintenance [2], [3], [4], [5]. Furthermore, since the CAN protocol is broadcast in nature and based on message priority, it is also vulnerable to Denial-of-Service (DoS) attacks. Initially, such attacks were mounted by transmitting high-priority messages [6]. Recent work shows that CAN buses are vulnerable to other types of DoS attacks, which are harder to detect [7].

To alleviate limitations of intra-car wired network, we propose in this work a hybrid wired/wireless network architecture for supporting intra-car communication. A wireless extension of the wired CAN bus can indeed enhance the electrical architecture flexibility of cars, by allowing the integration of new remote sensors without costly installation of physical wires or added connectors and complexity to the CAN buses. A wireless interface also provides redundancy in the case of a DoS attack on the wired interface. A key goal in this context is to achieve reliable and efficient delivery of packets from the sensors to a sink (ECU), a task also known as *data collection*.

In contrast to prior work on hybrid wired/wireless communication systems [8], [9], [10], [11], any node in our architecture can be connected to either the wired or wireless interfaces or *both* of them. Such a design raises several research issues. The first issue is how to implement routing. For instance, in the hybrid network of Fig. 1, packets destined from node 2 to the sink can be routed through either node 7, node 9, or node 12.

The second issue is how to implement load balancing. For instance, node 10 can communicate with the sink either on the wired interface or the wireless interface.

The third issue is how to deal with contention from other nodes and (possibly malicious) interferences. For instance, how should node 10 react if node 4 is contending on the wired link? And what happens if an adversary performs a DoS attack?

• *W. Si was with with the Division of Systems Engineering, Boston University; D. Starobinski is with the Department of Electrical and Computer Engineering, Boston University; M. Laifenfeld was with the Department of Electrical and Computer Engineering, Boston University Boston, MA 02215 USA*
E-mail: weisi@bu.edu; staro@bu.edu; moshel@spacegate.com.

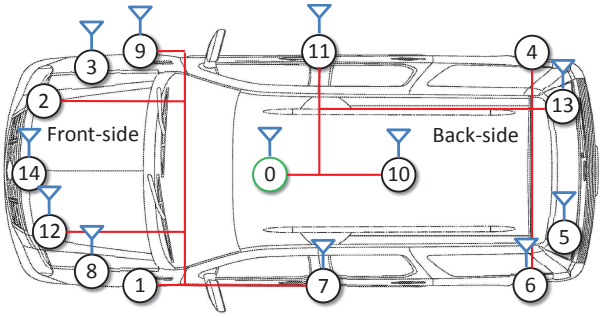


Fig. 1: A 15-node intra-car hybrid wired/wireless network. Each node is connected to either a wired interface or a wireless interface or both. The data packets of the sensor nodes (1-14) need to be delivered to the sink (node 0).

In light of these challenges, we define the following objectives for designing a collection protocol for hybrid intra-car networks:

- *Load balancing.* The protocol should utilize available interfaces and balance packet transmissions over the interfaces based on link conditions (e.g., bandwidth and congestion level) in order to achieve a certain optimization objective. In our case, the objective is throughput-optimality (see discussion below).
- *Routing.* In the absence of a direct communication link between a sensor node and the sink, the protocol should deliver the packets of the sensor node in a multi-hop fashion.
- *Robustness.* The protocol should achieve reliable data collection even when link qualities degrade (e.g., due to contention, interferences, or DoS attacks).
- *Backward-compatibility.* The protocol should not require the replacement of existing technology (e.g., CAN buses) in vehicles.

To address these issues, we propose a new data collection protocol for hybrid intra-car networks, called *Hybrid-BCP*. Hybrid-BCP belongs to the class of the state-of-the-art *backpressure* algorithms [12], [13], which have theoretically been proven to be throughput-optimal. This property means that all the queues in the network remain stable (i.e., do not grow indefinitely) and packets get delivered to the sink, as long as this is theoretically feasible. From a practical point of view, throughput-optimality is closely related to the reliability of packet delivery and network robustness. Indeed, a throughput-optimal protocol can better sustain traffic congestion or adversarial conditions affecting link qualities than a protocol that is not throughput-optimal.

We implement Hybrid-BCP on a real testbed, composed of CAN and ZigBee transceivers, and evaluate its performance. Our testbed experiments demonstrate the load balancing and routing functionalities of Hybrid-BCP. The results show that Hybrid-BCP improves throughput under DoS attacks on the CAN bus by a factor of 10. They also show that Hybrid-BCP is robust to jamming attacks on wireless links.

We further implement Hybrid-BCP in ns-3 for the purpose of simulating a larger network. We compare Hybrid-BCP with a tree-based collection protocol, which we refer to as *Hybrid-Collection Tree Protocol (Hybrid-CTP)*. Hybrid-CTP is based on the popular CTP protocol [14] and relies on the computation and update of end-to-end routing metrics at each node.

For the simulations, we use real RSSI (received signal strength indication) traces collected in an intra-car environment [15]. The simulation results demonstrate that Hybrid-BCP achieves higher throughput than Hybrid-CTP if both protocols use the same power transmission. The results also show that Hybrid-BCP maintains high performance for safety-critcal sensors that are connected to the same bus as the sink.

We summarize the contributions of this paper as follows:

- We design a new protocol, Hybrid-BCP, for data collection in intra-car hybrid wired/wireless networks.
- We prove the throughput-optimality of an idealized version of Hybrid-BCP.
- We build a real testbed for evaluating the performance of Hybrid-BCP. The tests demonstrate the load balancing and routing functionalities of Hybrid-BCP and its resilience to DoS attacks.
- We implement Hybrid-BCP and Hybrid-CTP in the ns-3 simulator, and compare their performance in terms of reliability for different transmission powers.

The rest of the paper is organized as follows. Section 2 reviews related work on hybrid wired/wireless networks, load balancing algorithms for multiple interfaces, and collection protocols. Section 3 describes the Hybrid-BCP protocol and its software implementation. Section 4 presents our throughput analysis. Section 5 and 6 provide performance evaluation of Hybrid-BCP in testbed experiments and simulations, respectively. Finally, Section 7 concludes the paper and discusses future research directions.

2 Related Work

2.1 Hybrid wired/wireless networks

Much of the existing work on hybrid wired/wireless networks assumes that all the devices (except for bridges or relays) are connected to either a wired interface or a wireless interface but not both.

For instance, [8] implements a hybrid wired/wireless network for greenhouse control and management using CAN and ZigBee transceivers. In that system, the central controller and a number of wireless bridges are connected to a bus. The bridges receive data from wireless sensors and forward them to the controller. The work in [9] conducts a feasibility study of a hybrid wired/wireless network implementation based on Ethernet and Bluetooth. In the implementation, sensors

have either a wired or a wireless interface while the sinks are connected to a bus. A bridge node communicates between the wireless nodes and the wired nodes. The work in [16] describes the design of a CAN-to-RF bridge for automotive environments.

Similar hybrid network structures can be found in [10], [11], where wireless nodes communicate with wired nodes through access points. In the hybrid wired/wireless models of [17], [18], a number of base stations are interconnected with high-bandwidth wired links and they serve as relays for the wireless nodes.

The issue of supporting two interfaces first raises the challenge of selecting the right interface for each packet in order to achieve an optimization objective. Beyond that, the addition of a wireless interface should not add significant protocol overhead or come at the expense of increased latency for safety-critical messages.

2.2 Load balancing

There exist several protocols for aggregating bandwidth and performing end-to-end load balancing. These protocols are implemented at the transport layer or above, and rely on protocols at lower layers to provide the routing functionality.

For instance, Multipath TCP (MPTCP) [19] uses multiple TCP paths to increase the throughput of data transfer. The earliest delivery path first (EDPF) [20] estimates the packet delivery time on several paths and schedules packets on the path with the shortest delivery time. The work in [21] adds to EDPF by incorporating transmission rates and losses in the estimation of the delivery time of packets. Other algorithms based upon EDPF include [22], [23], [24].

Different from the above work, Hybrid-BCP provides a joint load balancing and routing solution.

2.3 Multi-channel multi-interface wireless networks

There have been several works on designing channel assignment and routing protocols for multi-channel multi-interface wireless networks [25], [26], [27]. For example, J-CAR [25] selects the interface with the lowest load, and assigns to that interface the channel with the lowest amount of interference. The routing component of J-CAR is based on AODV.

Our work differs from prior works in that the interfaces considered in prior works have the same communication media while we focus on a hybrid heterogeneous multi-interface communication scheme. Although prior works consider channel interference in the protocol design, they do not provide demonstration of robustness against attacks while we do so through testbed experiments. In addition, we provide guarantees on the throughput performance of the Hybrid-BCP protocol through theoretical analysis.

2.4 Collection protocols

Collection protocols are routing protocols designed specifically for routing data from sensor nodes to a central collection node. There exist two well-known collection protocols in wireless sensor networks. The first one is the Collection Tree Protocol (CTP) [14]. CTP establishes a minimum-cost routing tree where the cost on each link equals the expected number of transmissions on that link (ETX).

The other one is the Backpressure Collection Protocol (BCP) [12]. BCP derives from backpressure routing algorithms [13], which achieve optimal throughput. With BCP, nodes independently make routing decisions based on local information. Routing decisions are made on a per-packet basis rather than on a pre-computed path. The work in [12] shows that BCP achieves higher throughput and reliability than CTP under dynamic network conditions (e.g., in the presence of external sources of interferences). Since Hybrid-BCP is built upon BCP, we briefly review how BCP operates in the next section.

Algorithm 1 BCP

1: Compute backpressure weight $w_{i,j}$ for each neighbor j
2: Find the neighbor j^* such that $j^* = \arg\max_j w_{i,j}$
3: **if** $w_{i,j*} > 0$ **then**
4: Transmit a packet to j^*
5: Update $\overline{ETX}_{i\to j^*}$ and $\overline{\mathcal{R}}_{i\to j^*}$
6: **else**
7: Wait for a reroute period and go to line 1
8: **end if**
9: Go to line 1

2.5 Backpressure Collection Protocol (BCP)

BCP does not calculate end-to-end routes. Rather it relies on a distributed computation of *backpressure weights*. Each node maintains a backpressure weight for each of its neighbors, based on the link quality (noise floor, packet collisions, etc.) and the differential of the queue lengths. For each incoming packet, a node selects the neighbor with the highest positive backpressure weight as the next hop. If all the backpressure weights are negative, then the node stores the packet in its queue and waits until one of the backpressure weights becomes positive.

Specifically, let $\mathcal{Q}_i$ represent the backlog (i.e., number of packets stored) at node i. Let $\overline{\mathcal{R}}_{i\to j}$ be the estimated link rate from i to j and let $\overline{ETX}_{i\to j}$ be an estimate of the average number of transmissions needed to successfully transmit a packet over the link. According to the routing policy of BCP, node i calculates the backpressure weight for each neighbor j as follows:

$$w_{i,j} = (\mathcal{Q}_i - \mathcal{Q}_j - V \cdot \overline{ETX}_{i\to j}) \cdot \overline{\mathcal{R}}_{i\to j},$$

where V is a trade-off parameter.

The routing decision (i.e., the selected next hop for the current packet) is determined by finding the neighbor j^* with the highest weight. Node i then makes the forwarding decision: if $w_{i,j^*} > 0$, then the packet is forwarded to node j^*, *else the packet is held until*

the metric is recomputed. In other words, if the weights for all neighbor nodes are zero or negative, the node will do nothing but wait until the next recomputation (after a *reroute period*). A pseudo-code of BCP is given in Algorithm 1.

BCP aims to minimize the expected number of packet transmissions (ETX) while guaranteeing network stability. The parameter V ($V \geq 1$) represents the weight on minimizing ETX in the optimization problem.

BCP estimates $\overline{ETX}$ based on an exponential moving weighted average formula. Whenever a new sample of ETX is obtained, $\overline{ETX}$ is updated as follows: $\overline{ETX}_{new} = \alpha\overline{ETX}_{old} + (1-\alpha)ETX$. The default value of α is 0.9. The link rate is calculated as the reciprocal of the packet transmission time (the time elapsing from the first transmission to the reception of an ACK), and the estimated link rate $\overline{\mathcal{R}}$ is updated according to an exponential moving weighted average formula similar to that used for $\overline{ETX}$.

In contrast to BCP, Hybrid-BCP is backward-compatible with the existing CAN technology and incrementally deployable (i.e., not every node needs to be hybrid). Moreover, Hybrid-BCP adds diversity and is robust to DoS attacks on either one of the media. Since BCP uses only the wireless medium, it does not have such robustness.

3 HYBRID-BCP

In this section, we describe the protocol design of Hybrid-BCP and its software implementation.

3.1 Protocol design

Hybrid-BCP can be viewed as two BCP algorithms running in parallel, with one algorithm handling the wired interface (e.g., CAN) and the other one handling the wireless interface (e.g., ZigBee).

Next, we describe the handler of interface l, where $l \in \{W, WL\}$ (W represents the wired interface and WL represents the wireless interface). Let $\overline{\mathcal{R}}^l_{i\to j}$ be the estimated link rate from i to j over interface l and let $\overline{ETX}^l_{i\to j}$ be an estimate of the average number of transmissions needed to successfully transmit a packet over the interface. Note that $\overline{ETX}^l_{i\to j}$ captures the effects of packet collisions and interferences. The interface handler of node i calculates the backpressure weight for each neighbor j on interface l as follows:

$$w^l_{i,j} = (\mathcal{Q}_i - \mathcal{Q}_j - V \cdot \overline{ETX}^l_{i\to j}) \cdot \overline{\mathcal{R}}^l_{i\to j}.$$

Let j^* denote the neighbor with the highest weight on the wired interface, i.e., $j^* = \arg\max_j w^W_{i,j}$. Let k^* denote the neighbor with the highest weight on the wireless interface, i.e., $k^* = \arg\max_k w^{WL}_{i,k}$.

A higher backpressure weight represents a link of higher quality and a neighbor with less backlog. A necessary condition for the wired interface handler to transmit a packet to neighbor j^* is that $w^W_{i,j^*} > 0$. When both the wired and wireless interface handlers are idle, an additional condition is that the weight of the wired interface is the larger one, i.e., $w^W_{i,j^*} \geq w^{WL}_{i,k^*}$. If one of these conditions is not satisfied, then the wired interface handler waits for the next computation of backpressure weights. Similar conditions apply for the wireless interface handler. Algorithm 2 provides a pseudo-code of Hybrid-BCP and Table 1 summarizes its scheduling procedure. Thus, Hybrid-BCP performs load balancing by selecting a link interface and next hop for each outgoing packet based on current estimation of the quality of each link and backlog at each possible next hop.

Algorithm 2 Hybrid-BCP

procedure WIRED_INTERFACE_HANDLER
  Wire_busy ← false
  while $\mathcal{Q}_i > 0$ **do**
    Compute the backpressure weight $w^W_{i,j}$ for each neighbor j on the wired link
    Find the neighbor j^* such that $j^* = \arg\max_j w^W_{i,j}$
    if $w^W_{i,j^*} > 0$ **and** (Wireless_busy = true **or** $w^W_{i,j^*} \geq w^{WL}_{i,k^*}$) **then**
      Wire_busy ← true
      Transmit one packet to j^* over the wired interface
      Update $\overline{ETX}^W_{i\to j^*}$ and $\overline{\mathcal{R}}^W_{i\to j^*}$
      Wire_busy ← false
    else
      Wait for a reroute period
    end if
  end while
end procedure

procedure WIRELESS_INTERFACE_HANDLER
  Wireless_busy ← false
  while $\mathcal{Q}_i > 0$ **do**
    Compute the backpressure weight $w^{WL}_{i,k}$ for each neighbor k on the wireless links
    Find the neighbor k^* such that $k^* = \arg\max_k w^{WL}_{i,k}$
    if $w^{WL}_{i,k^*} > 0$ **and** (Wire_busy = true **or** $w^{WL}_{i,k^*} > w^W_{i,j^*}$) **then**
      Wireless_busy ← true
      Transmit one packet to k^* over the wireless interface
      Update $\overline{ETX}^{WL}_{i\to k^*}$ and $\overline{\mathcal{R}}^{WL}_{i\to k^*}$
      Wireless_busy ← false
    else
      Wait for a reroute period
    end if
  end while
end procedure

3.2 Software implementation

The software implementation of Hybrid-BCP consists of a routing engine, a wired forwarding engine, a wireless forwarding engine and a beacon controller (see Fig. 2).

w^W_{i,j^*}	w^{WL}_{i,k^*}	Operation
> 0	≤ 0	Transmit the next packet to neighbor j^* on the wired link.
≤ 0	> 0	Transmit the next packet to neighbor k^* on the wireless link.
≤ 0	≤ 0	The next packet is not transmitted.
> 0	> 0	If both interface handlers are idle, the next packet is scheduled on the link with the larger weight. If one of the interface handlers is busy, the next packet is transmitted on the interface which is idle.

TABLE 1: Packet transmission scheduling of Hybrid-BCP.

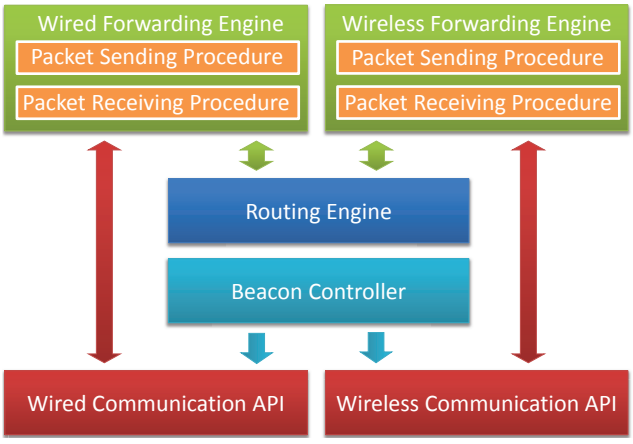


Fig. 2: Software architecture of Hybrid-BCP.

The routing engine is responsible for calculating the backpressure weights for each neighbor and interface. It updates and maintains the routing table.

The forwarding engine is responsible for scheduling packet transmissions and handling packet receptions. It is further composed of a packet sending procedure and a packet receiving procedure: the packet sending procedure runs the interface handler described in Algorithm 2, while the packet receiving procedure handles ACK packets and provides information for the routing engine to update the routing table.

The forwarding engine also keeps a count of transmissions for each packet. When the packet sending procedure transmits a packet on the interface, it waits to receive an ACK from the next hop until an *ACK timeout*. If an ACK is not received before the timeout, the packet sending procedure retransmits the packet on the interface.

Hybrid-BCP utilizes *beacon messages* to propagate backpressure information from a node to its neighbors. The beacon controller is responsible for broadcasting beacon messages on all available interfaces.

3.2.1 Protocol header

The protocol header of Hybrid-BCP contains the necessary information for the protocol to operate. The format of the protocol header is shown in Listing 1.

The Hybrid-BCP header consists of six fields: `origin`, `originSeqNo`, `bcpBackpressure`, `nextHop`, `lastHop`, and `type`. A data packet is identified by its source node ID, represented by `origin`, and its packet ID, represented by `originSeqNo`. The `bcpBackpressure` field is used to contain the backpressure information of the sender of a packet. Whenever other nodes overhear a packet, they extract the backpressure information of the sender and update the routing table. The `nextHop` field is set by the packet transmitter and when a neighbor receives the packet, it will check this field to determine whether it is the destined next hop. The `lastHop` field is used to record the ID of the node who transmits the packet. The `type` field represents the type of the packet.

```
struct hybrid_message_t
{
    byte origin;
    UInt16 originSeqNo;
    byte bcpBackpressure;
    byte nextHop;
    byte lastHop;
    byte type;
}
```

Listing 1: The protocol header of Hybrid-BCP.

There are three types of packets in Hybrid-BCP:

- Data packet – The data packet contains the data that should be delivered to the sink and is identified by its `origin` and `originSeqNo` fields;
- Beacon packet – The beacon packet contains the backpressure information of the sender. Neighbors of the sender who receive the beacon packet can update the backpressure of the sender in the routing table;
- ACK packet – The ACK packet is sent back by the receiver to the transmitter to indicate that the receiver has received the data packet. The ACK packet contains the same `origin` and `originSeqNo` fields as the corresponding data packet.

Note that the beacon packet is specifically used for broadcasting the backpressure information while other two types of packets also carry the backpressure information.

3.2.2 Packet sending procedure

In the packet sending procedure of the wired forwarding engine, the node first finds out the best neighbor (i.e., with the highest backpressure weight) on the wired bus. It then transmits a packet to the neighbor if the transmission conditions (in Algorithm 2) are satisfied. During the procedure, the forwarding engine also records the number of transmissions on the wired link for each packet. The flow chart of the packet sending procedure is shown in Fig. 3.

The forwarding engine uses a *sending queue* to store the packets that wait to be transmitted. The push and pop operations of the sending queue follow the last-in-first-out (LIFO) policy.

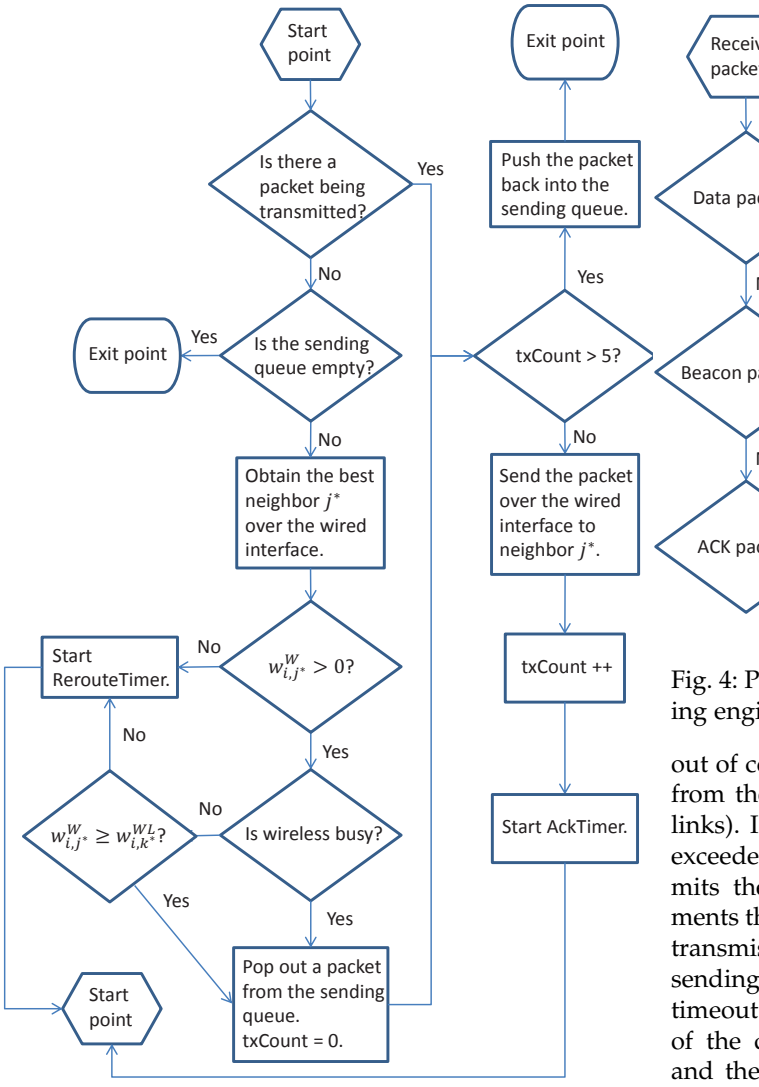


Fig. 3: Packet sending procedure of the wired forwarding engine.

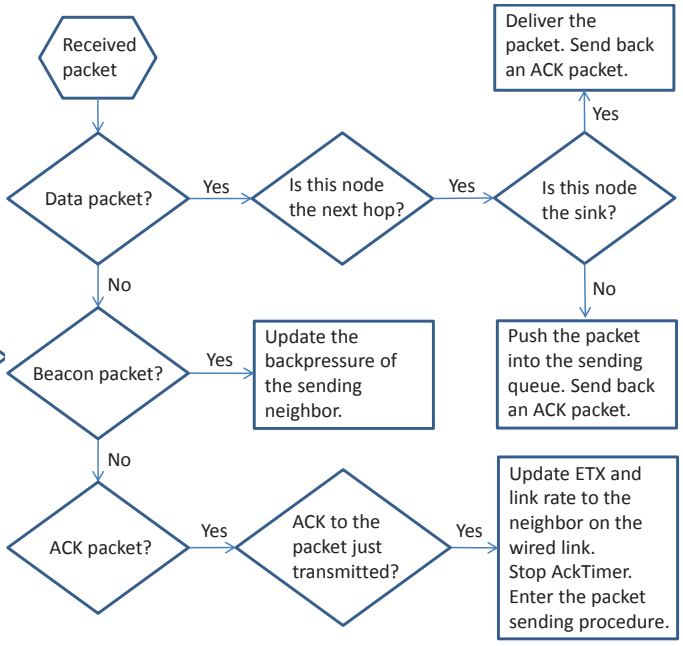


Fig. 4: Packet receiving procedure of the wired forwarding engine.

In the beginning of the packet sending procedure, the forwarding engine first checks whether there is a packet just transmitted but not ACKed yet. If there are no such packets and the sending queue is not empty, then the forwarding engine requests the routing engine to calculate the backpressure weights for the neighbors on the wired links. If the transmission conditions for the backpressure weights are not satisfied, the forwarding engine starts the reroute timer, and re-enters the packet sending procedure after a reroute period. Otherwise, a packet is popped out from the sending queue and the forwarding engine sets its next hop to the neighbor with the highest backpressure weight.

Next, if the transmission count (`txCount`) of the packet (a new packet, or a packet just transmitted but not ACKed) has reached a limit (five by default in our implementation), the packet is pushed back into the sending queue. The aim of having a limit on the transmission count is to prevent trying to constantly transmit a packet to neighbor which might have reached out of communication range (e.g., due to disconnection from the wired bus or intensive noise on the wireless links). If the transmission count of the packet has not exceeded the limit, the forwarding engine then transmits the packet to the selected next hop and increments the transmission count. After finishing the packet transmission, the ACK timer is started and the packet sending procedure will be re-entered after an ACK timeout. During the ACK timeout, if the ACK packet of the data packet is received, the ACK timer stops and the forwarding engine enters the packet sending procedure again to check whether more packets need to be transmitted. As packet collisions and interferences get more severe, the transmission count of a packet will increase. Thus ETX in the calculation of backpressure weights captures the effects of packet collisions and interferences.

3.2.3 *Packet receiving procedure*

When receiving a packet from a neighbor node, the packet receiving procedure of the forwarding engine handles differently depending on the type of the packet. The packet receiving procedure of the wired forwarding engine is depicted in Fig. 4.

If the received packet is a data packet and the sink is the next hop of the data packet (by checking the `nextHop` field), the sink delivers the data packet. If it is a sensor node that receives the data packet as the next hop, the sensor node pushes the data packet into its own sending queue. For both sensor nodes and the sink, an ACK packet corresponding to the data packet is sent back to the data packet transmitter. If the packet is an ACK packet and it is the corresponding ACK packet to the data packet that was just transmitted, it indicates that the data packet has been successfully received by

the next hop. Then the ACK timer will be stopped and routing information (ETX and link rate) of the neighbor on the wired link will be updated. The forwarding engine then enters the packet sending procedure to check whether more packets need to be transmitted. If the packet is a beacon packet, the forwarding engine simply extracts the backpressure information from the beacon packet and provides it to the routing engine for updating the routing table.

4 THROUGHPUT-OPTIMALITY

In this section, we show that an idealized version of Hybrid-BCP, which we call Multi-Interface Dynamic Backpressure Algorithm (MIDBA), achieves optimal throughput performance. The proof uses techniques presented in [13]. Our work differs from prior works in that prior works consider only one interface while we consider multiple interfaces.

We consider a network with penalties. In our case, penalties correspond to transmissions. The goal is to design a network control algorithm that minimizes the time average penalties while guaranteeing *network stability*, i.e., the time average queue sizes do not go to infinity. The *capacity region* represents the set of arrival rates that can be stabilized by *any* algorithm. An algorithm is called *throughput-optimal* if it can stabilize the network for all the rates in the capacity region.

The derivation of MIDBA is closely related to the *Lyapunov drift*, the differential of the Lyapunov function between the current time slot and the next time slot. MIDBA is motivated by the Lyapunov Optimization Theorem, which states that if, at each time slot, the Lyapunov drift plus the penalty function is upper bounded by a quantity related to the queue sizes, the time average queue sizes will not go to infinity. The theorem suggests minimizing on the Lyapunov drift plus the penalty function at each time slot. Next we describe the network model, using notations similar to [13], and define the Lyapunov function and Lyapunov drift. The steps used to minimize the bound on the Lyapunov drift plus penalty lead to the design of MIDBA.

Assume that the network evolution is time slotted. The network consists of N nodes. At each time slot t, the network topology state is $S(t)$ (such as noise power, interface connection status and wireless link gains). The backlog of the nodes in the network is $\mathbf{Q}(t)$ (i.e., $[\mathcal{Q}_1(t), \mathcal{Q}_2(t), .., \mathcal{Q}_N(t)]$). The arrival process is $\mathbf{A}(t)$ (i.e., $[A_1(t), A_2(t), .., A_N(t)]$).

Assume that there are L interfaces at each node. We further assume that the communication media of the interfaces are independent and the control on different interfaces are therefore independent. The network controller can apply control $I^l(t) \in \mathcal{I}^l_{S(t)}$ on interface l (e.g., whether to transmit or not, and set the transmission power), which yields a link transmission rate function $C^l_{ab}(I(t), S(t))$.

At time t, node a transmits $\mathcal{R}^l_{ab}(t)$ packets to node b on interface l, where $0 \leq \mathcal{R}^l_{ab}(t) \leq C^l_{ab}(I(t), S(t))$. Assume that $A_i(t)$ and $\mathcal{R}^l_{ab}(t)$ are bounded, i.e., $0 \leq A_i(t) \leq A_{max}$, $0 \leq \mathcal{R}^l_{ab}(t) \leq R_{max}$. The cost of the communication is $g_{ab}(\mathcal{R}^l_{ab}(t))$. We use a linear cost function $\mathcal{R}^l_{ab}(t)P^l_{ab}(t)$, where $P^l_{ab}(t)$ is the price paid for transmitting a packet on the interface l. More specifically, in our scenario, $P^l_{ab}(t)$ represents the number of transmissions for a packet to be successfully received from node a to b on interface l. The goal of the network controller is to minimize the long-term time average of the cost function while guaranteeing the network stability.

The quadratic Lyapunov function is defined as follows:

$$L(\mathbf{Q}) = \sum_i \mathcal{Q}_i^2. \tag{1}$$

The Lyapunov drift is

$$\Delta(\mathbf{Q}(t)) = L(\mathbf{Q}(t+1)) - L(\mathbf{Q}(t)). \tag{2}$$

Then the Lyapunov drift plus the penalty is

$$\Delta(\mathbf{Q}(t)) + Vp(t), \tag{3}$$

where $p(t) = \sum_l \sum_{ab} \mathcal{R}^l_{ab}(t)P^l_{ab}(t)$ and V indicates how much emphasis is put on the cost function.

Next we state the Lyapunov Optimization Theorem (Theorem 5.4 in [13]).

Theorem 1 (Lyapunov Optimization Theorem [13])**.** *Suppose there are constants $B > 0$, $\epsilon > 0$, $V \geq 0$, p^* such that for all t and all possible vectors $\mathbf{Q}(t)$ the following drift-plus-penalty condition holds:*

$$\mathbb{E}\{\Delta(\mathbf{Q}(t)) + Vp(t) | \mathbf{Q}(t)\} \leq B + Vp^* - \epsilon \sum_{i=1}^{N} \mathcal{Q}_i(t). \tag{4}$$

Then for all $t > 0$ the time average penalty and time average queue sizes satisfy:

$$\frac{1}{t}\sum_{\tau=0}^{t-1}\sum_i \mathbb{E}\{\mathcal{Q}_i(\tau)\} \leq \frac{B + Vp^*}{\epsilon} + \frac{\mathbb{E}\{L(\mathbf{Q}(0))\}}{\epsilon t},$$

$$\frac{1}{t}\sum_{\tau=0}^{t-1}\mathbb{E}\{p(t)\} \leq p^* + \frac{B}{V} + \frac{\mathbb{E}\{L(\mathbf{Q}(0))\}}{Vt}.$$

The Lyapunov Optimization Theorem states that if the drift plus penalty is upper bounded by the right-hand side of (4), then the time average queue sizes will not go to infinity, and network stability is guaranteed. The Lyapunov Optimization Theorem motivates us to minimize on the Lyapunov drift plus the penalty. Next we first describe the backlog dynamic and then derive a bound on the Lyapunov drift plus the penalty.

The backlog dynamic at node i is described as follows:

$$\mathcal{Q}_i(t+1) \leq \max[\mathcal{Q}_i(t) - \sum_l \sum_b \mathcal{R}^l_{ib}(t), 0] + A_i(t) + \sum_l \sum_a \mathcal{R}^l_{ai}(t). \tag{5}$$

The backlog of node i at the next time slot is obtained by reducing the number of transmitted packets from the backlog at the current time slot and then adding

exogenous and endogenous packet arrivals. The backlog dynamic is described by an inequality because some neighbor a might transmit less data packets than the assigned transmission rate $\sum_l \mathcal{R}^l_{ai}(t)$ if its queue backlog is less than the transmission rate, i.e., $\mathcal{Q}_a < \sum_l \mathcal{R}^l_{ai}(t)$.

Then by squaring both sides of (5) and based on Lemma 4.3 of [13], we have

$$\Big(\mathcal{Q}_i(t+1)\Big)^2 \le \Big(\mathcal{Q}_i(t)\Big)^2 + \Big(\sum_l \sum_b \mathcal{R}^l_{ib}(t)\Big)^2 + \Big(A_i(t) + \sum_l \sum_a \mathcal{R}^l_{ai}(t)\Big)^2 + 2\mathcal{Q}_i(t)A_i(t) - 2\mathcal{Q}_i(t)\Big(\sum_l \sum_b \mathcal{R}^l_{ib}(t) - \sum_l \sum_a \mathcal{R}^l_{ai}(t)\Big). \quad (6)$$

Next by summing (6) over all nodes, we have

$$\sum_i \Big(\mathcal{Q}_i(t+1)\Big)^2 - \sum_i \Big(\mathcal{Q}_i(t)\Big)^2 \le B + 2\sum_i \mathcal{Q}_i(t)A_i(t) - 2\sum_l \sum_{ab} \mathcal{R}^l_{ab}(t)\Big(\mathcal{Q}_a(t) - \mathcal{Q}_b(t)\Big), \quad (7)$$

where $B = N(A^2_{max} + 2NLA_{max}R_{max} + 2N^2L^2R^2_{max})$.

After adding the penalty function to (7), the Lyapunov drift plus the cost is

$$\Delta(\mathbf{Q}(t)) + Vp(t) \le B + 2\sum_i \mathcal{Q}_i(t)A_i(t) - 2\sum_l \sum_{ab} \mathcal{R}^l_{ab}(t)\Big(\mathcal{Q}_a(t) - \mathcal{Q}_b(t) - VP^l_{ab}(t)\Big). \quad (8)$$

Minimizing the upper bound on the Lyapunov drift plus the cost is equivalent of minimizing the right-hand side of (8). Since the term B is constant and $2\sum_i \mathcal{Q}_i(t)A_i(t)$ is observed at each time slot, minimizing the right-hand side of (8) is further equivalent of maximizing:

$$\sum_l \sum_{ab} \mathcal{R}^l_{ab}(t)\Big(\mathcal{Q}_a(t) - \mathcal{Q}_b(t) - VP^l_{ab}(t)\Big). \quad (9)$$

Since the interfaces are independent, maximizing (9) is equivalent of simply solving L maximization problems, i.e., for each interface l, maximizing $\sum_{ab} \mathcal{R}^l_{ab}(t)(\mathcal{Q}_a(t) - \mathcal{Q}_b(t) - VP^l_{ab}(t))$. This yields the Multi-Interface Dynamic Backpressure Algorithm (MIDBA), as shown in Algorithm 3.

MIDBA falls under the category of backpressure algorithms, thus it has the following performance (Theorem 6.2 of [13]).

Theorem 2 (Algorithm Performance [13])**.** *Assume that the network topology state $S(t)$ is i.i.d. from slot to slot and there exists a value $\epsilon_{max} > 0$ together with a stationary randomized algorithm (choosing control variables $I^l(t)$, $\mathcal{R}^l_{ab}(t)$ based only on the current network topology state $S(t)$) such that for all t and all node i we have:*

$$\sum_l \sum_a \mathbb{E}\{\mathcal{R}_{ai}(t)\} + \epsilon_{max} + \mathbb{E}\{A_i(t)\} = \sum_l \sum_b \mathbb{E}\{\mathcal{R}_{ib}(t)\}.$$

Algorithm 3 MIDBA

1: For all links (a, b), calculate the backpressure weight:
$$W^*_{ab}(t) = \max[\mathcal{Q}_a(t) - \mathcal{Q}_b(t), 0].$$
2: For each interface l, choose $I^l(t) \in \mathcal{I}^l_{S(t)}$ to maximize:
$$\sum_{ab} C^l_{ab}(I^l(t), S(t))[W^*_{ab}(t) - VP^l_{ab}(t)].$$
3: Over each interface l and each link (a, b) such that $W^*_{ab}(t) > VP^l_{ab}(t)$, transmit $\mathcal{R}^l_{ab}(t) = C^l_{ab}(I^l(t), S(t))$ units of data (using idle fill if necessary).

Then under MIDBA, we have:

$$\limsup_{t\to\infty} \frac{1}{t}\sum_{\tau=0}^{t-1}\sum_i \mathbb{E}\{\mathcal{Q}_i(\tau)\} \le \frac{B + VP_{max}}{\epsilon_{max}},$$
$$\limsup_{t\to\infty} \frac{1}{t}\sum_{\tau=0}^{t-1} \mathbb{E}\{p(\tau)\} \le p^* + \frac{B}{V},$$

where P_{max} is a constant such that $p(t) \le P_{max}$ for all t and p^* is the optimal cost over all stationary randomized policies.

Theorem 2 states that whenever a stationary randomized algorithm stabilizes the network, MIDBA can also achieve network stability. Thus MIDBA can achieve any rate in the capacity region and is therefore throughput-optimal.

Hybrid-BCP is an asynchronous, distributed version of MIDBA with a wired interface and a wireless interface, i.e., $L = 2$. Under Hybrid-BCP, for each transmission chance, each node a maximizes $\sum_{l\in\{W,WL\}} C^l_{ab}(I^l(t), S(t))[W^*_{ab}(t) - VP^l_{ab}(t)]$ with $C^l_{ab}(I^l(t), S(t))$ and $P^l_{ab}(t)$ being replaced by $\overline{\mathcal{R}}^l_{a\to b}$ and $\overline{ETX}^l_{a\to b}$, respectively. Thus Hybrid-BCP inherits the high throughput performance of MIDBA.

5 Experiments

In this section, we demonstrate the load balancing and routing functionalities of Hybrid-BCP in the testbed. We also show that Hybrid-BCP can be used to protect against DoS attacks on the CAN bus and jamming attacks on the wireless links.

5.1 Performance metrics

Before presenting the experiments, we provide the definition of metrics for evaluating the performance of Hybrid-BCP.

Suppose a test lasts for T seconds. Let N denote the total number of generated packets. Let N_u denote the number of delivered packets, excluding packet duplicates, and let $\mathcal{S}_d$ represent the set of the uniquely delivered packets.

ACK timeout for CAN link	30 ms
ACK timeout for ZigBee link	80 ms
Reroute period	50 ms
Beaconing period	1500-2000 ms
Queue capacity	48

TABLE 2: Parameters in the implementation of Hybrid-BCP for the testbed.

The *delivery rate* is defined to be the percentage of packets that are delivered, i.e., $\frac{N_u}{N} \cdot 100\%$. The *throughput* is defined to be the number of unique packets delivered to the sink per second, i.e., $\frac{N_u}{T}$ pkts/sec. The delay of a packet D_i is defined as the time elapsing from its generation at the source node to its delivery at the sink. The *average delay* is calculated as $\frac{1}{N_u} \sum_{i \in \mathcal{S}_d} D_i$.

If a node is directly connected to the sink through both wired and wireless interfaces, the fraction of packets delivered through the wired link for this node is $\frac{N_W}{N_W + N_{WL}}$, where N_W and N_{WL} are the number of packets delivered though the wired and wireless links respectively.

5.2 Experimental setup

We build a hybrid CAN/ZigBee network to test Hybrid-BCP. We use VN1610 CAN interfaces [28], manufactured by Vector Informatik GmbH, as CAN transceivers. We use TelosB motes [29] as ZigBee transceivers. The CAN bus is configured to operate at the rate of 33,333 baud. The transfer rate of a ZigBee transceiver is 250 Kb/s.

To emulate a node (a sensor or an ECU), we use a laptop to which one or both types of transceivers are connected. Our testbed has two kinds of laptops: Lenovo X220 (Intel Core i5-2520@2.50GHz, Windows 7) and Lenovo B590 (Intel Premium Dual-Core 2020M@2.4GHz, Windows XP). A laptop runs a Windows Presentation Foundation (WPF) application [30] to manage the interfaces. Hybrid-BCP is implemented in C#, as a component of the WPF application.

The first set of tests is conducted on the networks A, B, and C, whose topologies are shown in Fig. 5. Fig. 6 shows the testbed setup of network C.

We choose the ACK timeout for a CAN/ZigBee link to be slightly larger than the round trip time (RTT) of the link under light load conditions. The RTT of a CAN link is around 15 ms and that of a ZigBee link ranges from 50 ms to 70 ms. The ZigBee link has a higher RTT than a CAN link because ZigBee is based on CSMA/CA (Carrier Sense Multiple Access / Collision Avoidance) while CAN is based on CSMA/CD (Carrier Sense Multiple Access / Collision Detection).

Every time a beacon packet is transmitted, Hybrid-BCP waits for a *beaconing period* to transmit a new beacon packet. The beaconing period is chosen to be sufficiently large so that beacon packets do not cause congestion on the links. It is also uniformly randomly selected within a range of possible values to avoid possible synchronization of beacon packets between different nodes and contention on the links. Table 2 lists the parameters used in the Hybrid-BCP implementation.

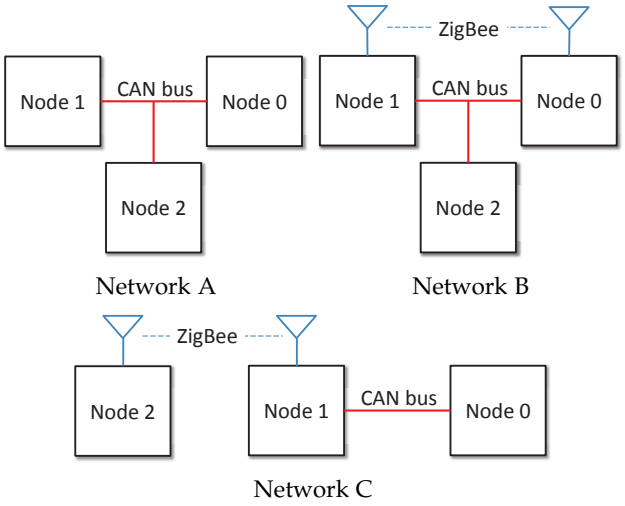


Fig. 5: The network topologies used for demonstrating the load balancing and routing functionalities of Hybrid-BCP on the testbed.

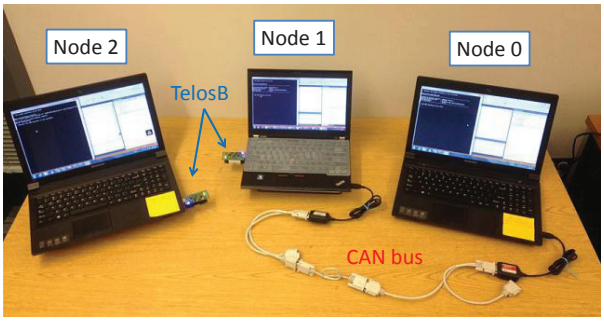


Fig. 6: Testbed setup for network C.

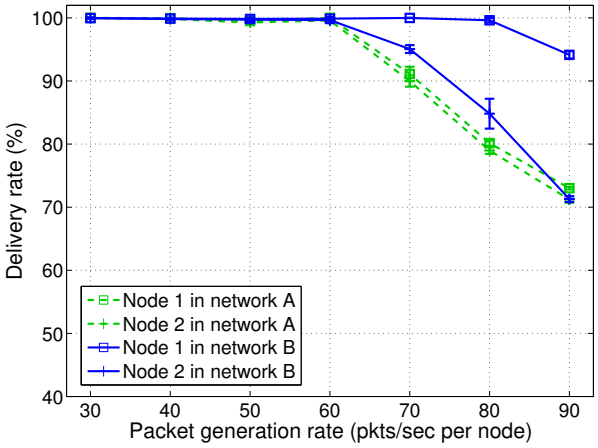
In the tests, each sensor node periodically generates data packets and transfers them to Hybrid-BCP, which delivers the packets to the sink. Sensor nodes generate packets at the same rate. Each test is run for five times to obtain an average and a 95% confidence interval for the metrics. Each run lasts from three to five minutes.

5.3 Load balancing

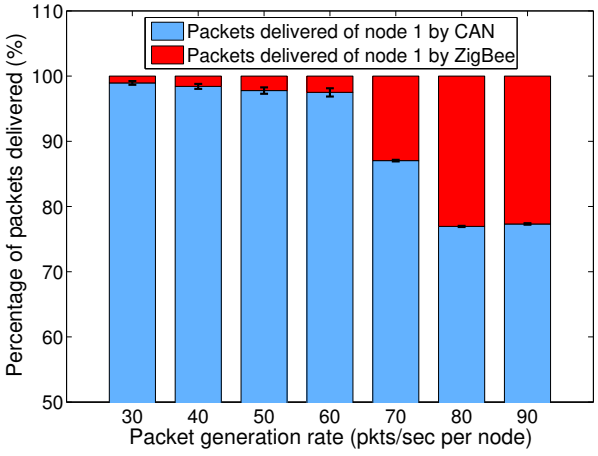
To demonstrate the load balancing functionality of Hybrid-BCP, we perform tests on network A (a CAN network) and network B (a hybrid CAN/ZigBee network).

Fig. 7(a) shows that at a packet generation rate of 80 pkts/sec, Hybrid-BCP improves the delivery rate of node 1 from 80.15% to 99.63% thanks to the additional wireless link. Moreover, the delivery rate of node 2 also improves, from 78.99% to 84.82%. This is because Hybrid-BCP transmits a fraction of packets of node 1 on the ZigBee link for the purpose of load balancing, hence reducing MAC contention on the CAN bus.

In network B, when the packet generation rate of node 1 is low, Hybrid-BCP schedules most of its packets on the CAN interface, as shown by Fig. 7(b). This is because the CAN link has a smaller RTT and thus a higher



(a) Delivery rate versus packet generation rate.



(b) Percentage of packets delivered over the CAN/ZigBee interfaces on network B.

Fig. 7: Performance of Hybrid-BCP on network A (CAN only) and network B (hybrid).

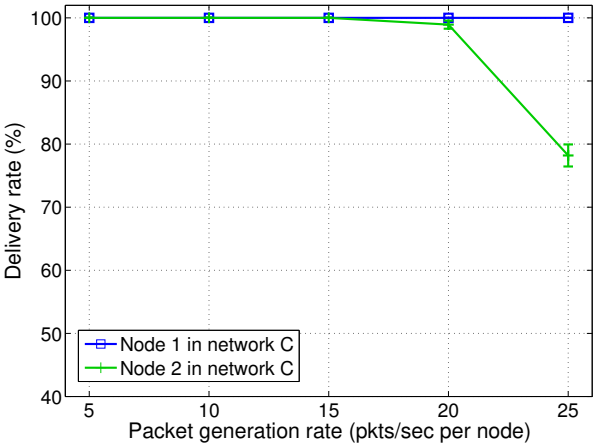
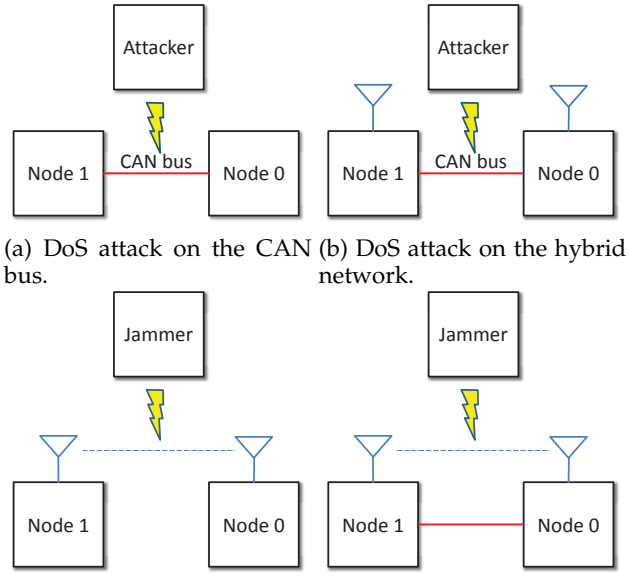


Fig. 8: Delivery rate versus packet generation rate of Hybrid-BCP on network C.



(a) DoS attack on the CAN bus.

(b) DoS attack on the hybrid network.

(c) Wireless jamming on the ZigBee link.

(d) Wireless jamming on the hybrid network.

Fig. 9: DoS attacks and wireless jamming attacks.

link rate than the ZigBee link. When the packet rate increases, the backlog of node 1 grows and Hybrid-BCP starts scheduling packet transmissions on the ZigBee link. When the packet rate reaches a certain threshold, the proportions of packets delivered through the CAN and ZigBee interfaces do not change further because both links are saturated.

At a packet generation of 90 pkts/sec, the CAN bus is saturated in both network A and network B. Thus, one would expect that the delivery rate of node 1 in network A should be the same as that of node 2 in networks A and B. However, the experimental results show that the former is slightly higher than the latter. We conjecture the cause of this discrepancy to be that node 1 has a faster CPU than node 2 (i.e., Lenovo X220 vs. Lenovo B590).

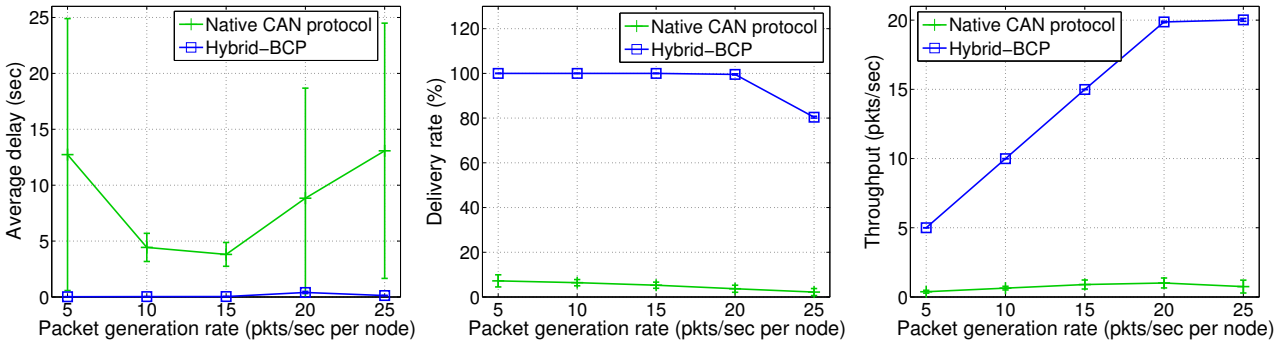
5.4 Routing

To demonstrate the routing functionality of Hybrid-BCP, we perform tests on network C. In network C, there is no direct communication link between node 2 and the sink. The packet delivery rates of node 1 and node 2 are plotted in Fig. 8. The results show that the delivery rate of node 2 is 98.93% at a packet generation rate of 20 pkts/sec. Thus, Hybrid-BCP can successfully route packets from node 2 to the sink via node 1. The ns-3 simulations in Section 6.2 demonstrate the routing functionality of Hybrid-BCP in a larger hybrid network.

5.5 Robustness

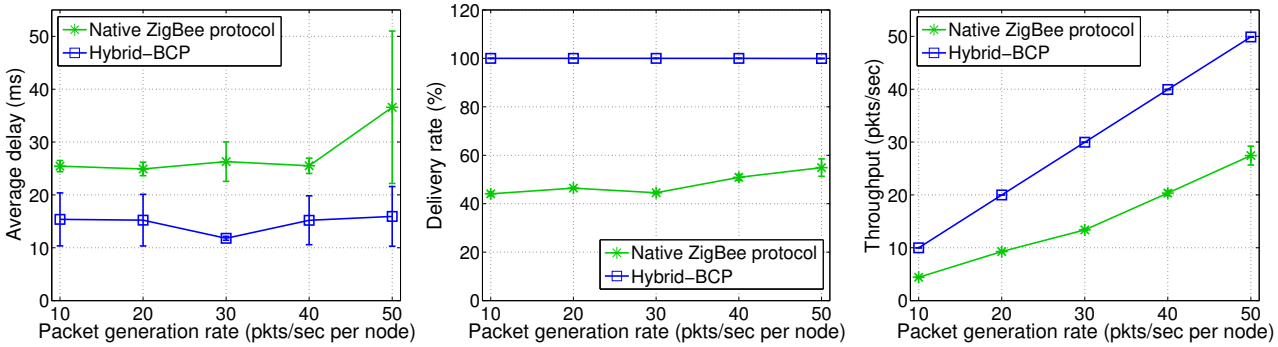
5.5.1 *DoS attacks on CAN*

The CAN protocol is a priority-based protocol: a high-priority message gets transmitted ahead of a low-priority message. Previous work in [31] shows that the injection of malicious CAN messages can be done by remotely compromising and controlling nodes on



(a) Average delay of the legitimate node versus its packet generation rate. (b) Delivery rate of the legitimate node versus its packet generation rate. (c) Throughput of the legitimate node versus its packet generation rate.

Fig. 10: Performance of the native CAN protocol and Hybrid-BCP under DoS attacks on the CAN bus. The attacker generates high-priority messages at a rate of 300 pkts/sec.



(a) Average delay of the legitimate node versus its packet generation rate. (b) Delivery rate of the legitimate node versus its packet generation rate. (c) Throughput of the legitimate node versus its packet generation rate.

Fig. 11: Performance of the native ZigBee protocol and Hybrid-BCP under wireless jamming attacks. The attacker generates high-priority messages at a rate of 100 pkts/sec.

the bus (via the radio, the tire pressure monitoring system or the Bluetooth component). In this section, we evaluate the effects of such DoS attacks on a *legitimate node* (or a non-compromised node).

We implement a DoS attack by having an attacker transmit high-priority messages on the CAN bus, at a rate of 300 pkts/sec. We compare Hybrid-BCP to the *native CAN protocol*, a transmission scheme in which a legitimate node periodically generates data packets and transfers them to the CAN transceiver. The performance of the native CAN protocol is tested in the network of Fig. 9(a) and the performance of Hybrid-BCP is tested in the network of Fig. 9(b).

Fig. 10(a) shows the average delay of packets by node 1 under the native CAN protocol and Hybrid-BCP. We see that the average delay of the native CAN protocol under the DoS attack reaches high values exceeding 3 sec (e.g., 3.81 sec at a packet generation rate of 15 pkts/sec by a legitimate node). The low-priority legitimate node is almost starved and must wait for a long time between each successful transmission. The delay experienced by a low-priority packet is also unpredictable and thus the variance of the average delay is very large. This result indicates that the DoS attack dramatically increases the MAC delay of a legitimate node.

On the other hand, under the same DoS attack, Hybrid-BCP achieves higher packet delivery rate and higher throughput than the native CAN protocol, as shown in Fig. 10(b) and 10(c). More specifically, Hybrid-BCP achieves a throughput of 19.87 pkts/sec at a packet generation rate of 20 pkts/sec by a legitimate node, more than ten times higher than that of the native CAN protocol. This gain is achieved because Hybrid-BCP measures a high ETX on the CAN link and schedules packet transmissions on the ZigBee link instead. These results demonstrate that Hybrid-BCP can protect the CAN bus against DoS attacks.

5.5.2 *Wireless jamming*

In the wireless jamming tests, a wireless jammer performs protocol-compliant jamming attacks on the ZigBee link. The jammer periodically generates packets and broadcasts them on the ZigBee link. In our tests, the rate the wireless jammer generates packets is 100 pkts/sec. We compare Hybrid-BCP with the *native ZigBee protocol*, which simply periodically generates data packets and sends them on the wireless link to the sink. The native ZigBee protocol is tested in the network of Fig. 9(c)

Ethernet data rate	4Mbps
Wi-Fi standard	802.11b
Wi-Fi mode	Ad hoc
Wi-Fi data rate	DSSS 11Mbps
Ethernet ACK timeout of Hybrid-BCP	1 ms
Wi-Fi ACK timeout of Hybrid-BCP	2 ms

TABLE 3: The parameters in ns-3 simulations of Hybrid-BCP.

and the hybrid wired/wireless protocol is tested in the network of Fig. 9(d).

Comparisons of performance between the native ZigBee protocol and those of Hybrid-BCP are shown in Fig. 11. The results show that Hybrid-BCP achieves lower delay and higher delivery rate than the native ZigBee protocol. For example, under wireless jamming, the delivery rate of the ZigBee protocol is at most 54.90% at a packet generation rate of 50 pkts/sec, while Hybrid-BCP achieves a delivery rate of 99.95%.

6 Simulations

In this section, we provide performance evaluation of Hybrid-BCP in ns-3 simulations. We compare Hybrid-BCP to a tree-based routing protocol in a simulated intra-car hybrid wired/wireless network. We also demonstrate the load balancing functionality of Hybrid-BCP under a higher wireless data rate.

6.1 Simulation setup

In the ns-3 simulations, we use the built-in Ethernet and Wi-Fi ns-3 libraries to simulate wired and wireless links. To mimic the behavior of CAN/ZigBee links, we configure the ns-3 simulations such that the Wi-Fi link has a higher rate but a larger RTT than the Ethernet link. Table 3 describes the simulation configuration and the parameters of Hybrid-BCP. The simulation configuration is used throughout the simulations except for Section 6.4, where we compare the performance of Hybrid-BCP for different Wi-Fi rates. Each simulation is run 15 times to obtain average and 95% confidence intervals of the measured metrics. The seed of the random number generator is set to a different value in each run.

While our simulations use the standard version of Ethernet, we note that in practice Original Equipment Manufacturers (OEM) may prefer to use a deterministic version, based on Time-Sensitive Networking (TSN) [32].

6.2 Routing and robustness

We use intra-car RSSI traces obtained from real experiments [15] to simulate a 15-node intra-car hybrid wired/wireless network. The trace records the link gains between each pair of nodes over time. The network topology is shown in Fig. 1. In this hybrid network, the sink is located on the driver seat, three sensors are placed in the engine compartment, four sensors are attached to the four wheels, three sensors are placed on passenger seats, and the rest is placed on the chassis.

Node	Avg. delay (ms)	Delivery (%)	Throughput (pkts/sec)	#Hops
1	42.08	97.79	58.64	3.05
2	43.49	97.74	58.64	3.06
3	60.54	91.86	55.08	2.44
4	0.50	99.98	59.82	1.00
5	2.33	99.26	59.39	1.00
6	1.91	99.98	59.97	1.00
7	3.92	97.93	58.76	2.01
8	36.43	93.08	55.80	2.98
9	31.90	97.94	58.65	2.43
10	1.01	99.98	59.88	1.00
11	0.69	99.98	59.85	1.00
12	31.72	98.01	58.64	3.03
13	0.79	99.98	59.85	1.00
14	43.25	93.23	55.81	2.88
Network	21.00	97.62	818.80	1.97

TABLE 4: Performance statistics of Hybrid-BCP in the simulated 15-node intra-car hybrid wired/wireless network. The radio power is -27 dBm and the packet generation rate is 60 pkts/sec per node.

We compare Hybrid-BCP to a tree-based routing protocol, that we call Hybrid-CTP. Hybrid-CTP is a variant of CTP tailored for a hybrid network (see the appendix for a description of this protocol). In the simulations, each sensor periodically generates and transfers data packets to the underlying protocol (Hybrid-BCP or Hybrid-CTP), which routes the packets to the sink.

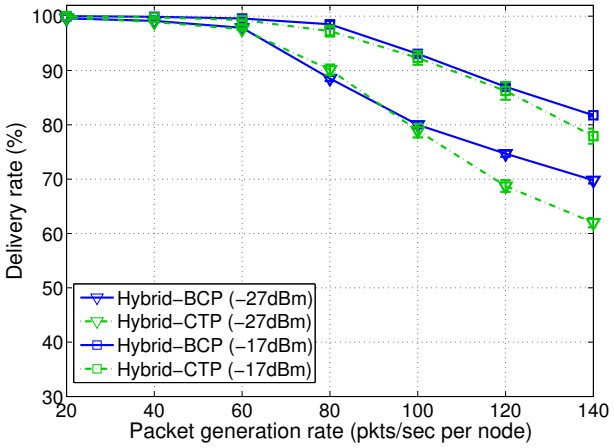
Fig. 12(a)-12(d) show the packet delivery rate, throughput, average hop count, and average delay of Hybrid-BCP and Hybrid-CTP under two different radio power levels. Fig. 12(a) and 12(b) show that Hybrid-BCP achieves higher delivery rate and throughput. At a packet generation rate of 140 pkts/sec, Hybrid-CTP achieves a throughput of 1206 pkts/sec while Hybrid-BCP achieves a throughput of 1354 pkts/sec, which improves by 12%.

Hybrid-CTP only considers the expected number of transmissions (ETX) while Hybrid-BCP minimizes both ETX and the queue backlog at different nodes. Hence, Hybrid-BCP is more responsive to congestion than Hybrid-CTP and achieves higher throughput.

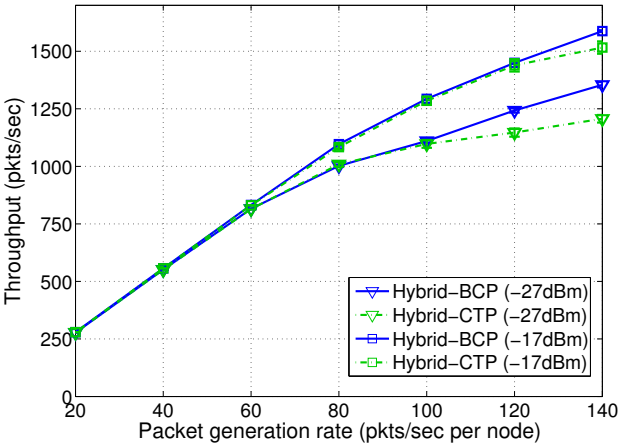
The routing functionality of Hybrid-BCP is further illustrated by statistics on the number of hops, as shown in Fig. 12(c). The figure shows that when the radio transmission power increases, the average number of hops decreases, which is to our expectation.

Fig. 12(d) shows that at low load, Hybrid-BCP has higher average delay than Hybrid-CTP, especially if the channel is lossy. As the load increases, the average delay of Hybrid-BCP decreases while that of Hybrid-CTP increases. This behavior of Hybrid-BCP is typical of backpressure algorithms (see [33] for more details and possible approaches to reduce delay at low load). As the network gets congested, Hybrid-BCP is more reponsive to network congestion than Hybrid-CTP. Hence, its average delay performance is superior in that regime.

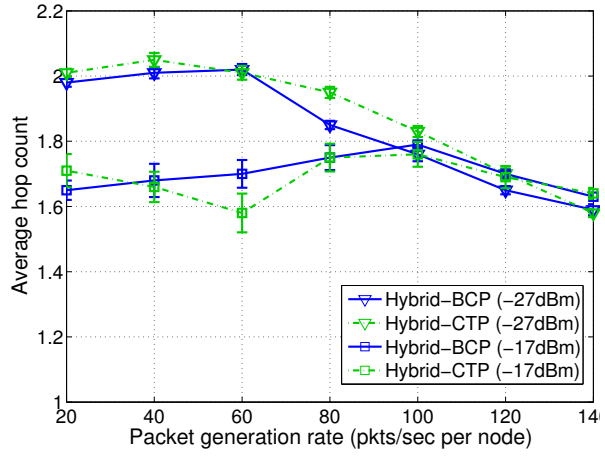
Table 4 shows per-node performance statistics of Hybrid-BCP according to the simulations. The radio power is -27 dBm and the packet generation rate is 60 pkts/sec at each node. We note that sensors that are connected to the same bus as the sink (i.e., nodes



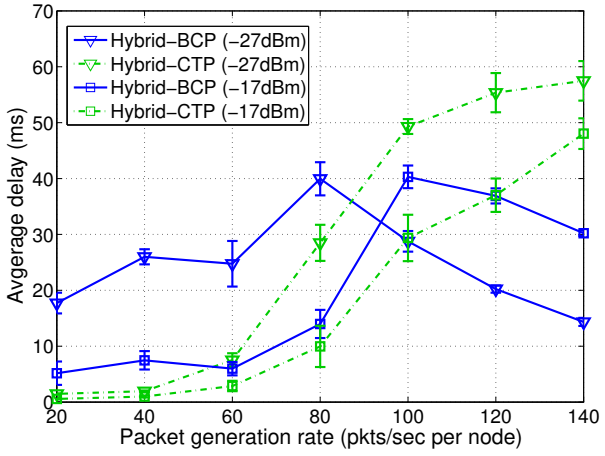
(a) Delivery rate versus packet generation rate.



(b) Throughput versus packet generation rate.



(c) Average hop count versus packet generation rate.



(d) Average delay versus packet generation rate.

Fig. 12: Performance of Hybrid-BCP and Hybrid-CTP in a simulated 15-node intra-car hybrid wired/wireless network. Hybrid-BCP achieves 12% higher throughput than Hybrid-CTP when the packet generation rate is 140 pkts/sec and the radio power is -27 dBm.

4, 6, 10, 11 and 13) achieve average delay smaller than 1 ms and reliability above 99% . As the distance (i.e., the number of hops) from the sink increases, the average delay increases and the throughput decreases. The results indicate that the proposed architecture can satisfy sensors with critical QoS requirements, if those sensors remain connected to the same bus as the sink.

6.3 Number of wireless interfaces

We study the impact of the number of wireless interfaces and their locations on the performance of Hybrid-BCP. We compare three different network topologies: (1) the original network as in Fig. 1; (2) the original network without wireless interfaces at nodes 12 and 13; (3) the original network without wireless interfaces at nodes 7, 12, and 13. The radio power is -27 dBm. The delivery rates of Hybrid-BCP on the three networks are shown in Fig. 13. The results show that removal of wireless interfaces from node 12 and 13 has small impact on the delivery rates, while removal of a wireless interface

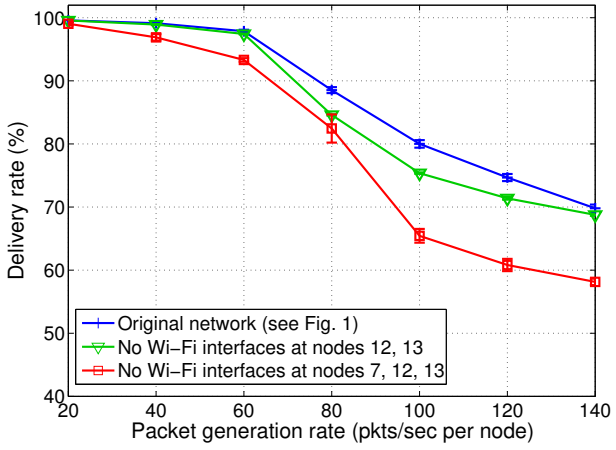


Fig. 13: Delivery rates of Hybrid-BCP with different number of wireless interfaces in the network.

from node 7 has larger impact. This is because packets

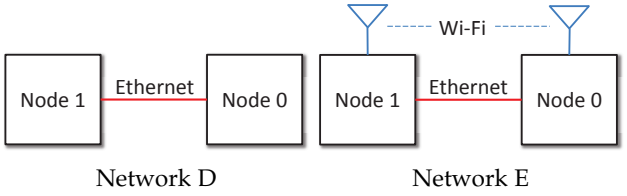


Fig. 14: The network topologies used for demonstrating the load balancing functionality of Hybrid-BCP in simulations.

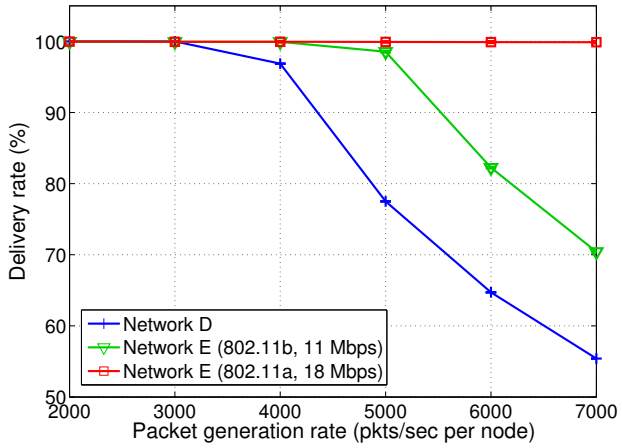


Fig. 15: Delivery rates of Hybrid-BCP on network D (Ethernet only) and network E (hybrid) in ns-3 simulations. The throughput improvement by load balancing of Hybrid-BCP is more significant as the wireless data rate gets higher.

from the front-side Ethernet bus are routed through nodes 7 and 9. With no wireless interface at node 7, the number of routing paths is reduced and so is the throughput.

6.4 Load balancing

We next show how Hybrid-BCP efficiently performs load balancing to aggregate more bandwidth, when such bandwidth is available. Hybrid-BCP is tested on network D (an Ethernet network) and network E (a hybrid Ethernet/Wi-Fi network), as shown in Fig. 14. We run simulations in the following three scenarios: (1) network D (no wireless link); (2) network E with Wi-Fi rate equal to 11 Mbps (802.11b); (3) network E with Wi-Fi rate equal to 18 Mbps (802.11a). The radio power is set to 10 dBm. The packet delivery rates under the three scenarios are plotted in Fig. 15. At a packet generation rate of 6,000 pkts/sec, Hybrid-BCP achieves a delivery rate of 61.71% when there is no wireless link. With an extra Wi-Fi link at the rate of 11 Mbps, the delivery rate is increased to 82.25%. The delivery rate is further increased to 99.90% when the Wi-Fi data rate is increased to 18 Mbps. The results show that Hybrid-BCP is able to take advantage of the additional capacity provided by the wireless links.

6.5 Wired DoS attacks and wireless jamming attacks

We implement the wired DoS attack by having an attacker on the bus. The simulation setup the same as Fig. 9(a) and 9(b). The simulation setup of wireless jamming attacks is the same as Fig. 9(c) and Fig. 9(d). The attacker (jammer) periodically generates and broadcasts packets on the bus (wireless link). We refer to the *native Ethernet (Wi-Fi) protocol* as the transmission scheme in which the legitimate node periodically generates and transfers data packets to the Ethernet (Wi-Fi) transceiver. The rates at which the wired attacker and the wireless jammer generate packets are set to 5,000 pkts/sec.

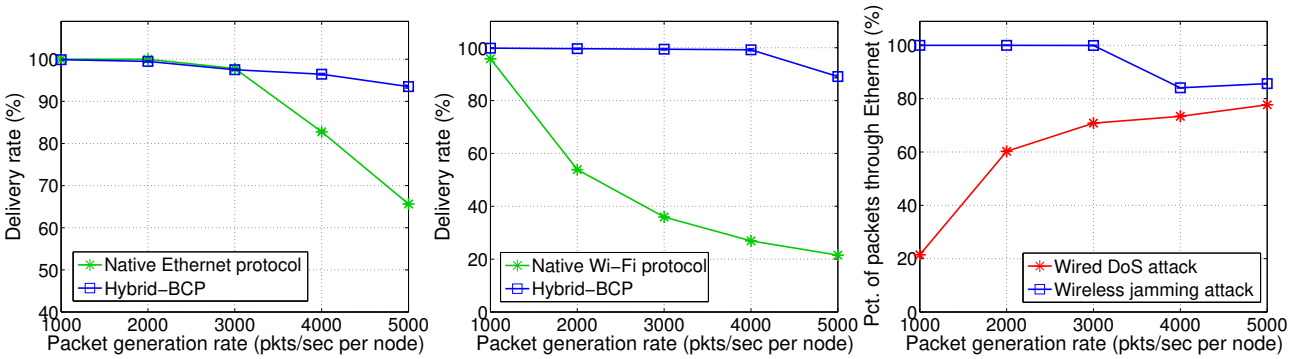
The performance of the native Ethernet (Wi-Fi) protocol and Hybrid-BCP under wired DoS attacks and wireless jamming attacks are shown in Fig. 16. Fig. 16(a) and 16(b) show that Hybrid-BCP achieves higher delivery rate than the native Ethernet protocol and the native Wi-Fi protocol under the attacks. Fig. 16(c) shows the percentage of packets delivered through the wired link by Hybrid-BCP under wired DoS attacks and wireless jamming attacks, respectively. It shows that at low load, Hybrid-BCP schedules most packets on the link which is not attacked. This indicates that Hybrid-BCP estimates the link qualities on both interfaces and chooses the link with better quality to transmit packets. As the traffic load increases, the non-attacked link becomes congested and Hybrid-BCP starts to schedule packet transmissions on the attacked link.

6.6 Attacks on intra-car network

We evaluate the performance of Hybrid-BCP and Hybrid-CTP on a large network under both wired DoS attacks and wireless jamming attacks. The network topology is the same as Fig. 1. A wired attacker periodically generates and broadcasts packets on the front-side Ethernet bus. A wireless attacker periodically generates and broadcasts packets on the wireless link. We assume that wireless paths exist between the wireless jammer and the wireless interfaces of nodes 9, 10, 13. The gain of each path is 60 dB. Both the wired attacker and the wireless jammer generate packets at 60 pkts/sec. The radio power is set to -27 dBm.

Fig. 17 shows the delivery rates of Hybrid-BCP and Hybrid-CTP under the attacks. At a packet generation rate of 100 pkts/sec per node, the attacks reduce the delivery rate of Hybrid-CTP from 78.92% to 68.25%, while the delivery rate of Hybrid-BCP is only reduced from 80.01% to 79.69%. These results demonstrate that Hybrid-BCP is more robust to DoS/jamming attacks than Hybrid-CTP.

At a packet generation rate of 120 pkts/sec (where the network is highly congested), the delivery rate of Hybrid-BCP under DoS attacks is higher than that of Hybrid-CTP under no DoS attack. This is because Hybrid-BCP is more responsive to network congestion than Hybrid-CTP.



(a) Delivery rate of the legitimate node versus its packet generation rate under a wired DoS attack.

(b) Delivery rate of the legitimate node versus its packet generation rate under a wireless jamming attack.

(c) Percentage of packets through the wired link of the legitimate node under the wired and wireless attacks.

Fig. 16: Performance of Hybrid-BCP under wired DoS attacks and wireless jamming attacks in ns-3 simulation.

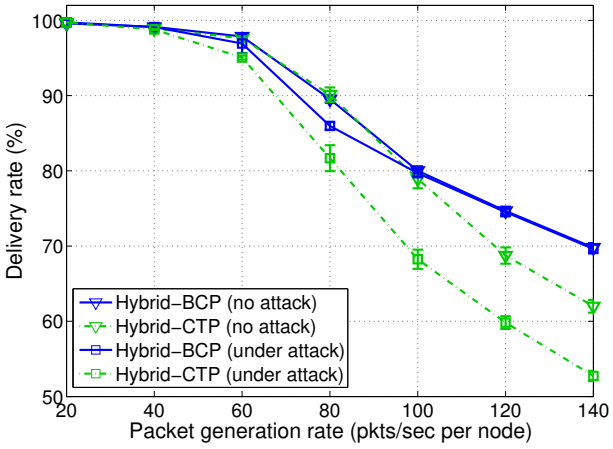


Fig. 17: Delivery rates of Hybrid-BCP and Hybrid-CTP under DoS attacks on a 15-node intra-car network.

7 Conclusion

In this paper, we designed and implemented Hybrid-BCP, a joint load balancing and routing solution for data collection in intra-car hybrid wired/wireless networks. Hybrid-BCP is backward-compatible with existing intra-car wired technologies since no modification of the CAN protocol is needed. Using Lyapunov optimization techniques, we proved the throughput-optimality of a centralized and slotted version of Hybrid-BCP.

We demonstrated the load balancing and routing functionalities of Hybrid-BCP in testbed experiments as proof of concept. We showed that Hybrid-BCP can be used to alleviate the impact of DoS attacks on the CAN bus. We also showed that Hybrid-BCP is robust to jamming attacks on the wireless links.

Through simulations of large intra-car hybrid networks based on dynamic RSSI traces collected from real experiments, we showed that Hybrid-BCP can relieve traffic congestion from the CAN bus, and achieves higher throughput than Hybrid-CTP. Hybrid-BCP maintains high performance for safety-critical sensors that are connected to the same bus as the sink while delivering packets of other sensors through multi-hop routing. Our results indicate that it is necessary for safety-critical sensors to have both interfaces to be robust while other sensors can be equipped with any interface.

Intra-vehicular systems have a natural requirement that data generated by some sensors have higher priority than data generated by others. For example, data from airbag system sensors should have higher priority than data from light sensors because the former are more safety-critical. Hence, it would be useful to incorporate packet prioritization into the load balancing and routing functionalities of Hybrid-BCP. One should also consider extending Hybrid-BCP to support data collection with multiple sinks. We leave these tasks as areas for future work.

Acknowledgements

This research was supported in part by NSF under grant CNS-1409053.

References

[1] W. Zeng, M. A. Khalid, and S. Chowdhury, "In-vehicle networks outlook: achievements and challenges," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 3, pp. 1552–1571, 2016.

[2] M. Hashemi, W. Si, M. Laifenfeld, D. Starobinski, and A. Trachtenberg, "Intra-car wireless sensors data collection: A multi-hop approach," in *VTC*, 2013.

[3] K. Pretz, "Fewer wires, lighter cars." http://theinstitute.ieee.org/benefits/standards/fewer-wires-lighter-cars, Apr 2013.

[4] BMW, "Wiring harness." http://www.bmw.com/com/en/insights/technology/technology_guide/articles/wiring_harness.html?source=index&article=wiring_harness.

[5] S. Blanco, "How does weight affect a vehicle's efficiency." http://www.autoblog.com/2009/10/29/greenlings-how-does-weight-affect-a-vehicles-efficiency/, Oct 2009.

[6] K. Koscher, A. Czeskis, F. Roesner, S. Patel, T. Kohno, S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, and S. Savage, "Experimental security analysis of a modern automobile," in *Security and Privacy (SP), 2010 IEEE Symposium on*, pp. 447–462, May 2010.

[7] A. Palanca, E. Evenchick, F. Maggi, and S. Zanero, "A stealth, selective, link-layer denial-of-service attack against automotive networks," in *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, pp. 185–206, Springer, 2017.

[8] O. Mirabella and M. Brischetto, "A hybrid wired/wireless networking infrastructure for greenhouse management," *Instrumentation and Measurement, IEEE Transactions on*, vol. 60, pp. 398–407, Feb 2011.

[9] D. Miorandi and S. Vitturi, "Hybrid wired/wireless implementations of profibus dp: A feasibility study based on ethernet and bluetooth," *Comput. Commun.*, vol. 27, pp. 946–960, June 2004.

[10] Y. Sun and E. M. Belding-Royer, "Application-oriented routing in hybrid wireless networks," in *Communications, 2003. ICC'03. IEEE International Conference on*, vol. 1, pp. 502–506, IEEE, 2003.

[11] L. Seno, S. Vitturi, and F. Tramarin, "Experimental evaluation of the service time for industrial hybrid (wired/wireless) networks under non-ideal environmental conditions," in *Emerging Technologies Factory Automation (ETFA), 2011 IEEE 16th Conference on*, pp. 1–8, Sept 2011.

[12] S. Moeller, A. Sridharan, B. Krishnamachari, and O. Gnawali, "Routing without routes: the backpressure collection protocol," in *IPSN*, 2010.

[13] L. Georgiadis, M. J. Neely, and L. Tassiulas, *Resource Allocation and Cross-Layer Control in Wireless Networks*. Foundations and Trends in Networking, 2006.

[14] O. Gnawali, R. Fonseca, K. Jamieson, M. Kazandjieva, D. Moss, and P. Levis, "Ctp: An efficient, robust, and reliable collection tree protocol for wireless sensor networks," *ACM Trans. Sen. Netw.*, vol. 10, pp. 16:1–16:49, Dec. 2013.

[15] W. Si, M. Hashemi, L. Xin, D. Starobinski, and A. Trachtenberg, "Teacp: A toolkit for evaluation and analysis of collection protocols in wireless sensor networks," *Network and Service Management, IEEE Transactions on*, vol. 12, pp. 293–307, June 2015.

[16] J. Park, C. Lee, J.-H. Park, B.-c. Choi, and J. G. Ko, "Poster: Exploiting wireless CAN bus bridges for intra-vehicle communications," in *Vehicular Networking Conference (VNC), 2014 IEEE*, pp. 111–112, IEEE, 2014.

[17] B. Liu, Z. Liu, and D. Towsley, "On the capacity of hybrid wireless networks," in *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, vol. 2, pp. 1543–1552 vol.2, March 2003.

[18] A. Zemlianov and G. de Veciana, "Capacity of ad hoc wireless networks with infrastructure support," *Selected Areas in Communications, IEEE Journal on*, vol. 23, pp. 657–667, March 2005.

[19] A. Ford, C. Raiciu, M. Handley, S. Barre, U. C. D. Louvain, and J. Iyengar, "Architectural guidelines for multipath tcp development", draft-ietf-mptcp-architecture-05 (work in progress," 2011.

[20] K. Chebrolu and R. Rao, "Communication using multiple wireless interfaces," in *Wireless Communications and Networking Conference, 2002. WCNC2002. 2002 IEEE*, vol. 1, pp. 327–331 vol.1, Mar 2002.

[21] G. Liu, X. Zhou, and G.-x. Zhu, "A scheduling algorithm for maximum throughput based on the link condition in heterogeneous network," *Journal Communication and Computer*, 2007.

[22] J. C. Fernandez, T. Taleb, M. Guizani, and N. Kato, "Bandwidth aggregation-aware dynamic qos negotiation for real-time video streaming in next-generation wireless networks," *Trans. Multi.*, vol. 11, pp. 1082–1093, Oct. 2009.

[23] K. Chebrolu, B. Raman, and R. R. Rao, "A network layer approach to enable tcp over multiple interfaces," *Wirel. Netw.*, vol. 11, pp. 637–650, Sept. 2005.

[24] A. L. Ramaboli, O. E. Falowo, and A. H. Chan, "Bandwidth aggregation in heterogeneous wireless networks: A survey of current approaches and issues," *Journal of Network and Computer Applications*, vol. 35, no. 6, pp. 1674 – 1690, 2012.

[25] H. S. Chiu, K. L. Yeung, and K.-S. Lui, "J-car: An efficient joint channel assignment and routing protocol for ieee 802.11-based multi-channel multi-interface mobile ad hoc networks," *IEEE Transactions on Wireless Communications*, vol. 8, pp. 1706–1715, April 2009.

[26] P. Kyasanur and N. H. Vaidya, "Routing and interface assignment in multi-channel multi-interface wireless networks," in *IEEE Wireless Communications and Networking Conference, 2005*, vol. 4, pp. 2051–2056 Vol. 4, March 2005.

[27] P. Kyasanur and N. H. Vaidya, "Routing and link-layer protocols for multi-channel multi-interface ad hoc wireless networks," *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 10, pp. 31–43, Jan. 2006.

[28] Vector Informatik GmbH, "VN1600 Network Interface for CAN, LIN, K-Line, J1708 and IO." http://vector.com/vi_vn1600_en.html.

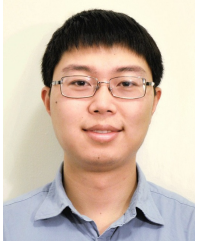
[29] MEMSIC Inc., "WSN Nodes - TPR2420." http://www.memsic.com/wireless-sensor-networks/TPR2420.

[30] Microsoft Corporation, "Windows Presentation Foundation." https://msdn.microsoft.com/en-us/library/ms754130(v=vs.100).aspx.

[31] S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, S. Savage, K. Koscher, A. Czeskis, F. Roesner, T. Kohno, *et al.*, "Comprehensive experimental analyses of automotive attack surfaces.," in *USENIX Security Symposium*, San Francisco, 2011.

[32] S. Tuohy, M. Glavin, C. Hughes, E. Jones, M. Trivedi, and L. Kilmartin, "Intra-vehicle networks: A review," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, pp. 534–545, April 2015.

[33] W. Si, D. Starobinski, M. Hashemi, M. Laifenfeld, and A. Trachtenberg, "Channel sensitivity of lifo-backpressure: Quirks and improvements," *IEEE Transactions on Control of Network Systems*, vol. 3, pp. 192–205, June 2016.



Wei Si received B.S. degree in Information Engineering from Shanghai Jiao Tong University, Shanghai, China, in 2010. He received his Ph.D. in Systems Engineering from Boston University in 2016. His research interests include routing protocols for wireless sensor networks and disruption tolerant networks, data synchronization algorithms and queueing theory.

David Starobinski is a Professor of Electrical and Computer Engineering at Boston University, with a joint appointment in the Division of Systems Engineering. He is also a Faculty Fellow at the U.S. DoT Volpe National Transportation Systems Center. He received his Ph.D. in Electrical Engineering from the Technion - Israel Institute of Technology, in 1999. In 1999-2000, he was a visiting post-doctoral researcher in the EECS department at UC Berkeley. In 2007-2008, he was an invited Professor at EPFL (Switzerland). Dr. Starobinski received a CAREER award from the U.S. National Science Foundation (2002), an Early Career Principal Investigator (ECPI) award from the U.S. Department of Energy (2004), the 2010 BU ECE Faculty Teaching Award, and best paper awards at the WiOpt 2010 and IEEE CNS 2016 conferences. He was an Associate Editor of the IEEE/ACM Transactions on Networking from 2009 to 2013. His research interests are in wireless networking, network economics, and cybersecurity.

Moshe Laifenfeld received his BSc ('92), MSc ('98) and Ph.D ('08) from the Technion, Tel-Aviv University and Boston University, respectively, all in electrical and computer engineering. After a joint post-doctoral position at MIT and Boston University, he joined General Motors R&D, focusing on in-vehicle wireless communications. In his past, Moshe led the algorithms development of a 3rd generation UMTS transceiver, and held several R&D positions in medical devices start-ups.

Algorithm 1 Hybrid-CTP

procedure WIRED_INTERFACE_HANDLER
  while $\mathcal{Q}_i > 0$ **do**
    Compute the $ETX_{i,j}^{W}$ for each neighbor j on the wired link
    Find the neighbor j^* such that $j^* = \arg\min_j ETX_{i,j}^{W}$
    if $ETX_i^{W^*} < ETX_i^{WL^*} + T$ **then**
      Transmit one packet to j^* over the wired interface
      Update $\overline{ETX}_{i\to j^*}^{W}$ and ETX_i
    else
      Wait for a reroute period
    end if
  end while
end procedure

procedure WIRELESS_INTERFACE_HANDLER
  while $\mathcal{Q}_i > 0$ **do**
    Compute the $ETX_{i,k}^{WL}$ for each neighbor k on the wireless link
    Find the neighbor k^* such that $k^* = \arg\min_k ETX_{i,k}^{WL}$
    if $ETX_i^{WL^*} < ETX_i^{W^*} + T$ **then**
      Transmit one packet to k^* over the wireless interface
      Update $\overline{ETX}_{i\to k^*}^{WL}$ and ETX_i
    else
      Wait for a reroute period
    end if
  end while
end procedure

APPENDIX A
PROTOCOL DESIGN OF HYBRID-CTP

In this section, we describe Hybrid-CTP, a variant of CTP designed for data collection in hybrid wired/wireless networks.

The same as Hybrid-BCP, Hybrid-CTP has two procedures handling the wired and wireless interfaces, respectively. Suppose for node i, node j is a neighbor on interface l, where $l \in \{W, WL\}$ (W represents the wired interface and WL represents the wireless interface). Let $\overline{ETX}_{i\to j}^{l}$ denote an estimate of the average number of transmissions needed to successfully transmit a packet from i to j over interface l.

Each node i records its end-to-end *path cost* to the sink, denoted by ETX_i. To obtain ETX_i, node i first calculates the path cost through interface l via node j as follows:

$$ETX_{i,j}^{l} = ETX_j + \overline{ETX}_{i\to j}^{l}.$$

The minimum path cost from node i to the sink node through interface l is $ETX_i^{l^*} = \min_j ETX_{i,j}^{l}$.

Then node i calculates its path cost to the sink by:

$$ETX_i = \min\{ETX_i^{W^*}, ETX_i^{WL^*}\}.$$

The path cost information is propagated to neighbors by beacon messages, the same as the backpressure information in Hybrid-BCP. The sink broadcasts path cost of zero.

In Hybrid-CTP, the wired interface handler schedules a packet transmission when $ETX_i^{W^*} < ETX_i^{WL^*} + T$, where T is a positive number (set to 0.5 in our implementation). Similarly, the wireless interface handler schedules a packet transmission when $ETX_i^{WL^*} < ETX_i^{W^*} + T$. Therefore, when $ETX_i^{W^*}$ is much smaller than $ETX_i^{WL^*}$, only the wired interface handler will schedule packet transmissions. This could happen either when the wireless link quality is bad or when all the neighbors on the wireless link select this node as their next hop. When $ETX_i^{W^*}$ and $ETX_i^{WL^*}$ are close to each other, both interface handlers will transmit packets. Algorithm 1 provides a pseudo-code of Hybrid-CTP.