Pangu: Towards a Software-Defined Architecture for Multi-function Wireless Sensor Networks

Junchen Guo, Yuan He, Xiaolong Zheng
School of Software and TNLIST, Tsinghua University
{juncguo, heyust, zhengxiaolong.zxl}@gmail.com

***Abstract*—Software-defined networking (SDN) is deemed as a promising direction to offer generalizability of wireless sensor networks (WSN). To introduce SDN into WSNs, however, means a series of non-trivial challenges due to the wireless and ad-hoc nature of WSNs. In this paper, we present our study towards a software-defined architecture for multi-function wireless sensor networks. Our proposal called Pangu is built upon the opportunistic routing protocol stack and introduces the concept of modality properties of sensor nodes. It enables centralized network control over a WSN while preserving the flexibility of underlying ad-hoc routing. We tackle the critical problems of the architecture design by presenting three essential components of Pangu. Moreover, we implement Pangu on a real-world testbed and evaluate it with various experiments.**

I. Introduction

As a technology to bridge the physical world and the cyberspace, wireless sensor network (WSN) has developed a lot in the past two decades. Today the WSN area faces an awkward dilemma. On one side, there have been countless research results regarding almost every building block of the technology, showing good readiness towards various practical uses. But on the other side, in spite of the ever increasing and prosperous demands, real WSN applications and systems are still far restricted to very few scenarios [1][2].

Poor generalizability is the major obstacle. In conventional WSNs, the upper-level applications are deeply coupled with the lower-level network stacks. The development of a WSN application always means considerable cost and low efficiency. The structure as well as the performance of underlying networks cannot smoothly evolve as the application demands are modified. Deep coupling also makes it extremely difficult to figure out a one-size-fits-all solution to be standardized and applied to a wide range of scenarios.

In the process of tackling the above problem, we are enlightened by the concept of software-defined networking (SDN). SDN, which separates the control plane from the data plane, brings flexibility of applications and simplicity of management into the wired network, especially the data center network [3][4]. The OpenFlow [5] protocol enables network users to centrally manage the network from a higher perspective through the standardized and expressive interfaces.

What about incorporating SDN into the design of a WSN? That is a natural question many researchers would think about. Working out a satisfactory answer, however, is a non-trivial task. We highlight the following comparisons between a WSN and a wired network, to identify the critical challenges of designing a software-defined architecture for WSNs.

- *Connectivity.* Unlike wired connections, wireless connections are intrinsically localized. A controller of a WSN cannot be directly connected to every sensor node at no expense, and vice versa. This introduces considerable delay and uncertainty into the processes of network state collection and control dissemination.
- *Consistency.* Network-wide consistency of configurations is a key element of SDN, but consistency is relatively hard to maintain in WSNs, due to unreliable wireless transmissions and unpredictable delay of multi-hop forwarding.
- *Data flow.* Unlike the scattered data flows in a wired network, data flows in a WSN generally converge to the sink node(s). Managing WSN data flows, however, is not simplified at all. Not only the routing direction, but also the data delivery performance, should be considered.

Given the above facts, we realize that completely separating the control plane from the data plane as traditional OpenFlow-based SDNs do is not well-suited for WSNs. Centralizing the entire control logic in remote controller(s) means a great deal of network overhead for frequent state report and control dissemination. Moreover, the delay of the multi-hop duty-cycled communication makes it difficult to modify flow entries to match the network fluctuation.

Consequently, in this work, we present Pangu[1], a software-defined architecture that enables network users to centrally control WSNs over the robust and efficient opportunistic routing stack. Pangu is designed to enhance WSNs with much more flexibility and also preserve the ad-hoc nature of WSNs. Especially, we tailor Pangu to a multi-function WSN that serves multiple applications with different types of sensor data.

To be more specific, Pangu defines the forwarding behaviors of the nodes over an opportunistic routing (OR) stack [6][7]. The main novelty is that we introduce the concept of *modality* properties (*node modality* and *modal filter*) of the sensor nodes. A node marks its node modality on the network-layer header of every packet it generates. The modal filter of a node declares a collection of modalities it should forward. Thus, modality is a representation of grouping. Identified by it, nodes could be grouped by the data types, the capacities of their hardware, or even the priorities of sensing tasks, etc.

In the tradition OR, upon receiving a packet, the node that first wakes up decides relaying or dropping it, only by checking whether himself has a routing progress, e.g.

[1]In Chinese mythology, Pangu was an ancient god who remoulded the earth by creating mountains and rivers, as what this work does to the OR in WSNs.

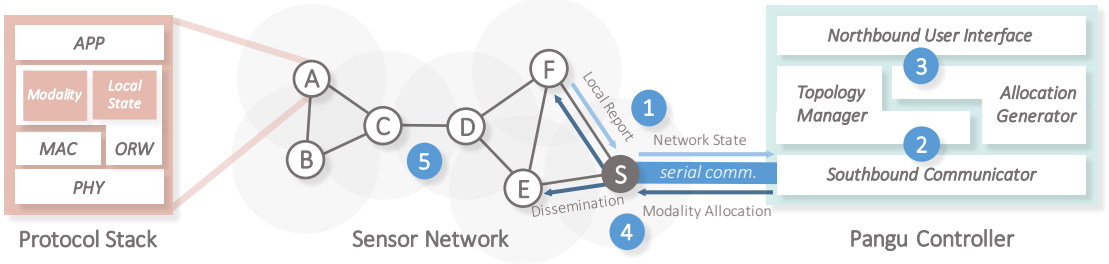


Fig. 1. **An overview of our software-defined architecture for WSNs, Pangu.** The underlying OR protocol stack is enhanced with modality module and local state module to achieve central control. The nodes communicate with the controller through the sink node.

a smaller expected transmission count (ETX) [8], over the last-hop node [7]. To alter forwarding behaviors, we propose modality matching as an additional condition. Therefore, by centrally defining modality properties of all sensor nodes, one can dynamically manage the network without digging into the underlying protocol stack. We choose OR for two reasons: 1) there are multiple potential forwarders of one packet in opportunistic routing, which provides the redundancy for pruning, and 2) in a receiver-dominated protocol, a node makes routing decisions only by considering received packets and its current state, which reduces the computation and simplifies the design.

Such a design has following potential advantages: First, defining modality properties draws an appropriate boundary between the centralized part and the distributed part of the control plane. Second, assigning modality properties to sensor nodes does not sabotage the flexibility of ad-hoc routing. Third, sensor functions and hardware capacities are easily available information for network users. Thus, they could manage a WSN from high-level perspectives simply by allocating modality properties. The contributions of our work are summarized as follows.

- We discuss how to split the control plane for a feasible software-defined architecture for WSNs, and propose a practical solution, Pangu, which enables central control over distributed routing (§III).
- In our design, Pangu is dedicated to addressing the problems of maintaining the global network state, generating a feasible modality allocation under certain application-level constraint(s) and disseminating the centralized control (§IV).
- We implement Pangu with TinyOS and a 50-node testbed. Then we conduct extensive experiments to evaluate the efficacy of Pangu and the network performance in terms of multiple metrics (§V).

II. Related Works

For decades, the exploration of providing flexible WSNs has never stopped. Network reprogramming is one the most representative works, which replace logical modules of sensor nodes just over the radio [9][10][11][12]. Nevertheless, this technique involves piles of interactions with the underlying system implementation, which is of high technical barriers. Differently, Pangu provides simple interfaces for network users to flexibly alter forwarding behaviors of nodes.

Another family of state-of-the-art works introduces role-based approaches [13][14][15], in which roles are interpreted from the perspective of upper-level applications. Kochhal *et al.* [13] introduce a distributed algorithm for assigning hierarchical roles, and Frank *et al.* [14] propose a framework with generic role specification as a programming abstraction. Based on these works, Miyazaki *et al.* [15] present a software-defined WSN that assigns application roles. However, Pangu defines routing roles of nodes, instead of application roles.

Recently, some works have extended SDN to WSN, trying to make WSN configurable by centrally defining forwarding behaviors of nodes [16][17][18]. Among those works, SDN-WISE [18] is the first OpenFlow-like prototype implemented on real world sensor nodes. By referring to a stateful flow table, the node in SDN-WISE can make forwarding decisions according to the centrally-defined flow rules and its local state machine. However, SDN-WISE neither leverages the ad-hoc nature of WSNs, or emphasizes critical problems such as the maintenance of global states and the dissemination of control instructions. Furthermore, the state machine involves deep packet parsing, which lacks abstraction for the application layer and is of high overhead.

In order to provide a practical SDN architecture for WSNs, the basic intuition is that we could only centralize one part of control logic and must preserve the other part on sensor nodes. Fibbing [19] gives us an exquisite example of how to feasibly draw this boundary. By introducing the concept of fake nodes, Fibbing can tell small lies to unmodified OSPF routers to accomplish desired alteration of underlying routing structure. In short, OpenFlow-like approaches completely define the underlying routing structure, while Fibbing just modifies parts of it. We believe that a feasible software-defined WSN should be even more conservative. Thus, Pangu just prunes the original routing structure into sub-nets of desired routes.

III. Pangu Design

A. Overview

Figure 1 presents an overview of Pangu. The nodes with an ORW stack [7] are centrally controlled by a controller. The communication between sensor nodes and the controller can be divided into two parts, the multi-hop wireless communication between sensor nodes and the sink node, and the

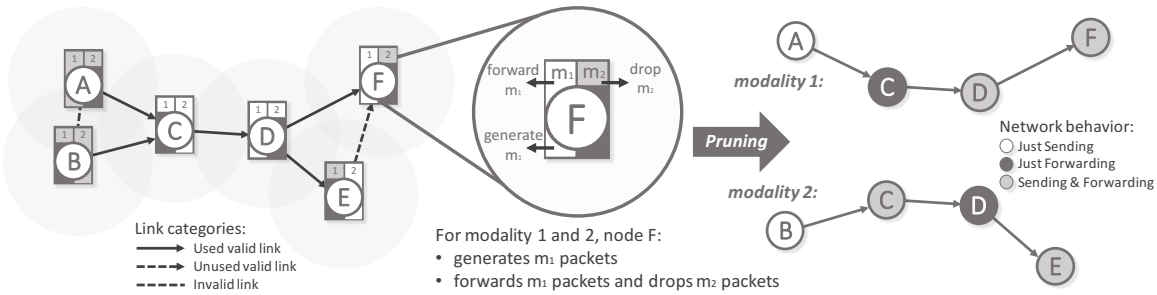


Fig. 2. **How Pangu prunes underlying routing structure with modality properties.** Suppose there are 2 modalities m_1, m_2 in the network. Left part of the figure shows one possible routing structure constructed by OR protocol, and one possible corresponding modality allocation of the nodes. Middle part details forwarding behaviors of node F under the allocation where $\mathrm{NM}(F) = m_1$ and $\mathrm{MF}(F) = \{m_1\}$. Right part shows the pruning results under current allocation and the different forwarding behaviors of the nodes in 2 paths.

serial communication between the sink node and the controller running on a PC. The controller contains 4 components, *Northbound User Interface*, *Allocation Generator*, *Topology Manager* and *Southbound Communicator*. It enables network users to centrally allocate modality properties to alter forwarding behaviors based on high-level demands.

The 5-step work flow of Pangu is shown in Figure 1:

1) **Local network state report:** The nodes maintain local network states in a distributed manner, and adaptively report to the controller.
2) **Global network state construction:** After local network states being parsed asynchronously by Southbound Communicator, Topology Manager should construct a global network state, in spite of the inestimable latency and the possible incompleteness.
3) **Modality properties allocation:** Northbound Interface presents the inferred global network state. And network users can either input their own modality allocation through the interface, or simply utilize the automatic Allocation Generator with specific performance demand(s).
4) **Control dissemination:** The allocation is encapsulated and sent to the sink node, and then been disseminated through the whole network. We ensure that no error occurs due to possible inconsistencies of local forwarding behaviors, and the system can be eventually consistent.
5) **Data collection:** Finally, the nodes generate and relay data packets according to the OR mechanism and modality properties.

B. Routing with Modality

Next, we describe how to change forwarding behaviors of the nodes by defining modality properties. The principles are:

- A node v has two centrally-defined modality properties, *node modality* $\mathrm{NM}(v)$ and *modality filter* $\mathrm{MF}(v)$.
- v should mark its current node modality $\mathrm{NM}(v)$ on the network-layer header of every packet it generates.
- If v is going to forward a packet p from a neighbor u, v should be the first to receive p and have a routing progress over u (OR conditions). Moreover, v's modal filter $\mathrm{MF}(v)$ must contain the modality marked on p (Modality requirement).

Consequently, a node with a non-empty modal filter might relay some specific packets, while a node with an empty modal filter would never relay any packet. Then we can just define modality properties of the nodes centrally, to modify their network behaviors.

The priority of OR conditions is always higher than that of the modality requirement, which means such centrally-controlled forwarding routes just belong to a subset of the underlying routing structure. Thus, the modality requirement is a method for pruning the OR routing structure. The advantages are as follows:

- Pruning the routing structure means that we don't have to track the status of all wireless links in real time.
- The efficiency of OR is preserved since a packet can still be relayed opportunistically by a remote node, only if that node satisfies the modality requirement.

Figure 2 presents how Pangu prunes local routing structure into two paths. Suppose there are 2 modalities m_1 and m_2 in the network. The annotation shows that for node F, $\mathrm{NM}(F) = m_1$, $\mathrm{MF}(F) = \{m_1\}$ in Figure . Since $\mathrm{NM}(F) = m_1$, node F marks m_1 on every packets it generates. Because $\mathrm{MF}(F) = \{m_1\}$, node F will only relay m_1 packets ($m_1 \in \mathrm{MF}(F)$) and drop m_2 packets ($m_2 \notin \mathrm{MF}(F)$). Thus, under this allocation, the nodes make forwarding decisions according to their modality properties and this local network is pruned into one path for m_1 data and another for m_2 data.

IV. Pangu Components

This section presents detailed designs of three main components of Pangu, which address the problems of maintaining the global network state (§IV-A), generating a feasible modality allocation under certain application-level constraint(s) (§IV-B) and disseminating the centralized control (§IV-C).

A. State Maintenance

The prerequisite for making centralized control is to get a real-time and complete global network state at the controller. However, the latency in a multi-hop wireless network, where nodes work in a duty-cycle mode, is inevitable. The constraint on network overhead and the limitation on payload length together make it unrealistic to report the complete local state. Meanwhile, the integrity of the global network state can not

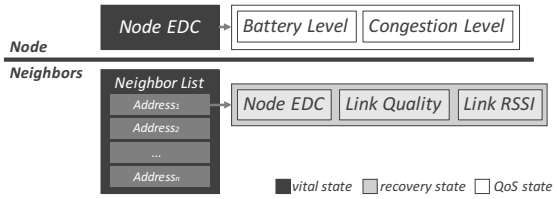


Fig. 3. **Content of local network state.**

be guaranteed due to the unpredictable packet loss in WSNs. Thus, making a feasible state maintenance strategy is one of the most essential problems we focus on.

The process of state maintenance has two alternating steps, *state report* and *state inference*. *State report*, running on the node side, decides what and when to report, while *state inference*, running on controller side, determines how to tolerate the possible incompleteness of the local network state.

State category. The local network state of a node contains two parts, the state of the node itself, including its EDC (Expected number of Duty-Cycled wake-ups, an improved version of ETX in OR [7]), the battery level, the congestion level, etc., and the state of its neighbors, including their addresses, EDCs, link ETXs, RSSIs, etc. Figure 3 shows 3 categories of local network states. *Vital state* including node EDC and its neighbor list is requisite for reconstructing the underlying opportunistic routing structure. *Recovery state*, including node EDC, link ETX and RSSI of neighbors, is redundant and used to tolerate possible incompleteness of *vital state*. All the other properties of the node, e.g. the battery level and the congestion level, etc., which enable the network users to making QoS decisions, are called *QoS state*.

State report. A node piggybacks its local state in the payload of packets. Reporting frequency is controlled by a period parameter T adaptively ranging from T_{min} to T_{max}. Initially, T is set to T_{min} during bootstrapping, in order to catch up with network fluctuations in this phase. Then, T increases additively until reaching T_{max}. During the process of additive increase, a node u computes the fluctuation parameter of its local network state:

$$f_u = \sum_{t=1}^{|\mathrm{NB}_u|} sgn(\mathrm{EDC}_u - \mathrm{EDC}_{\mathrm{NB}_u(t)} - \omega) \times 2^{t-1}$$

NB_u is the neighbor list of u, and $|\mathrm{NB}_u|$ refers to the length. $\mathrm{NB}_u(i)$ denotes the ith neighbor of u. Parameter ω is a constant value describing the cost of forwarding a packet over one hop [7]. $sgn(x) = \begin{cases} 1 & ,x \geq 0 \\ 0 & ,x < 0 \end{cases}$ is a sign function. As we can see, only when the fluctuation is huge enough to alter the underlying routing structure will f_u changes. Thus, Pangu uses f_u as an indicator, and multiplicatively decreases T when detecting a significant fluctuation.

State inference. Next, we describe how to utilize these asynchronous reports to infer a global network state. In a network of n sensor nodes, we denote the minimal global network state as an adjacent matrix $\mathcal{A} = (a_{ij})_{n\times n}$ and an EDC vector $\mathcal{E} = (e_i)_{1\times n}$, which are essential for centralized modality allocation. Here, we choose to use *vital state*and partial *recovery state* (neighbor EDCs) to infer $\mathcal{A}$ and $\mathcal{E}$.

To get $\mathcal{A}$ and $\mathcal{E}$, we introduce the concept of EDC matrix $\mathcal{M} = (m_{ij})_{n\times n}$ representing both two of them. $m_{ii} = e_i$ is the EDC of node i. If $m_{ij} > 0$ ($i \neq j$), it means that j is a neighbor of i, and i receives j's EDC, m_{ij}. If m_{ij} equals 0, it means that node i and j are not adjacent. Ideally, we should have a synchronous state where $\forall i \in [1,n], m_{ii} > 0$ and $m_{ji} = m_{ii}, \forall j \in \{j \mid m_{ji} > 0, j \in [1,n]\}$. However, the reports come asynchronously with unpredictable packet losses. Thus, the controller can only maintain $\mathcal{M}$ by processing these reports one by one, with the consideration of filling vacancies.

The report from node i ($\forall i \in [1,n]$) containing (i, EDC_i) and $\{(j, \mathrm{EDC}_j) | j \in \mathrm{NB}_i\}$ can be converted to a sparse report vector $\mathcal{R}_i = (r_k)_{1\times n}$, where $r_k = \begin{cases} EDC_k & ,k \in \{i\} \bigcup \mathrm{NB}_i \\ 0 & ,otherwise \end{cases}$. Then, upon receiving $\mathcal{R}_i$, we could simply replace $\mathcal{M}_i$, the ith row of $\mathcal{M}$, with it. Nevertheless, the order of receiving time on controller doesn't have to comply with the order of sending time on the node, due to the inestimable transmission delay. Thus, before replacing $\mathcal{M}_i$ with a newly-coming $\mathcal{R}_i$, we should first compare the sequence number of corresponding packets. The controller records q_{last}, the sequence number of the packet associated with the last replacing operation for each row. And it should abort current maintenance task if q_{rcv}, the sequence number of the packet containing $\mathcal{R}_i$, is not larger than q_{last}.

After the above replacing operation, we should decide whether to update m_{jj} with the value of r_j, $\forall j \in \mathrm{NB}_i$. The sequence number won't work, since the nodes are not synchronized. We additionally equip $\mathcal{M}$ with a *reference time* t_{ref}. $t_{ref}(i)$ stands for the referred sending time of the packet that updates m_{ii} (EDC_i). Supposing the packet is generated by node k, we can define

$$t_{ref}(i) = t_{update}(i) - EDC_k \times T_{node}$$

, where $t_{update}(i)$ is updating time and T_{node} represents the constant working period of the nodes. Then, we replace m_{jj} with r_j only when $t_{ref}(i) > t_{ref}(j)$.

The above strategy based on a transmission model leveraging the definition of EDC is concise but effective. However, more exquisite and complicated models could take link quality, link RSSI or the other user-specific recovery state into consideration to obtain a more meticulous global network state.

B. Modality Allocation

As a SDN architecture, Pangu shields underlying details of the network stack by abstracting nodes' forwarding behaviors, and provides the interface of configuring modality properties to manipulate such behaviors. Although the controller could hand over the task of modality allocation to network users as what traditional SDN does, we should have a better understanding of the allocation problem in some respects, such as how to generate a feasible allocation and how to satisfy performance requirements, etc.

From routes to a feasible allocation. With the EDC matrix $\mathcal{M}$ offered by topology manager, one can easily convert it to the underlying routing structure, which can be represented as a DAG $G(V, E)$. Then, we can introduce the necessity of a feasible allocation. After pruning G into a set of sub-nets $\{G_i(U_i \bigcup V_i, E_i)|i \in [1, k]\}$ (§III-B) with a feasible allocation, it is necessary to provide $\forall v \in V$ with at least one route to the sink in sub-net G_i, if $v \in V_i$ ($\mathrm{NM}(v) = m_i$). Consequently, assuming that $\forall v \in V$ is provided with a route set P_v that contains one or multiple routes from v to the sink, we can generate an allocation with the following steps:

1) Assign $\forall v \in V$ with one node modality $\mathrm{NM}(v) \in M = \{m_1, ..., m_k\}$, guided by upper-level multi-function applications.
2) For $\forall v \in V$, enumerate every route $p \in P_v$
3) Mark the modal filter $\mathrm{MF}(u)$ of node $u \in \{u|u \neq v,$ route p contains node $u\}$ with $\mathrm{NM}(v)$, that is, $\mathrm{MF}(u) = \mathrm{MF}(u) \bigcup \{\mathrm{NM}(v)\}$

Obviously, the easiest way to generate a feasible allocation is to uniformly set the modal filters of all nodes to M. Then, Pangu degrades to the ordinary ORW. Hence, to make a more proper allocation that can really satisfy upper-level requirements, it is necessary to bring in additional constraints for the network, e.g. end-to-end delay, throughput, load balance, etc.

Allocation under one constraint. End-to-end delay is one of the most common but essential performance requirements for WSNs. To achieve a lower expected end-to-end delay, we could generate a modality allocation that minimizes the maximum hop count from the nodes to the sink, which is often denoted as *network radius*. Next, we present a simplified algorithm to generate a minimum-radius allocation by enhancing breadth-first search (BFS):

1) Do BFS from the sink, and build a BFS tree $T_{\mathrm{BFS}}(V_{T_{\mathrm{BFS}}}, E_{T_{\mathrm{BFS}}})$
2) For an edge $e = (v, u) \notin E_{\mathrm{BFS}}$, if the hop count difference between v and u equals 1, add e to $E_{T_{\mathrm{BFS}}}$ and finally get a desired route graph G_{BFS}
3) For a node v and its children list CD_v, compute its modal filter recursively with $\mathrm{MF}(v) = \bigcup_{u \in \mathrm{CD}_v} (\mathrm{NM}(u) \bigcup \mathrm{MF}(u))$.

Step 2 aims at balancing load among the nodes who have the same hop count in T_{BFS}, without disturbing minimization of network radius.

Allocation under multiple constraints. The above algorithm is supposed to generate a flat routing graph, which makes backbone nodes with relatively small hop count very busy. To limit the traffic load of these nodes, the constraint on in-degrees of the nodes could be further brought in.

A few works have been working on solving the bounded-degree minimum-radius spanning tree (BDMRST) problem [20][21]. First, BDMRST is proved to be NP-hard. Then, for any instance of BDMRST with degree constraint Δ^*, these works propose (α, β)-bicriteria approximation algorithms to provide a spanning tree satisfying the following two conditions: 1) The degree of any nodes in the spanning tree is at most $\alpha + \Delta^*$. 2) The radius of the spanning tree is at most $\beta \cdot \mathrm{OPT}$, where OPT is the minimum possible radius of any spanning tree whose degree is bounded by Δ^* [21]. By improving this family of algorithms, that is $\alpha \to 0, \beta \to 1$, we can get a more and more accurate approximation of the optimal solution. Thus, we can apply these algorithms to build a spanning tree, and then recursively compute modal filters of the nodes.

The above two examples are common but simple. More constraints, such as minimum duty cycle, residual power limitation, QoS constraint, and multi-function priority etc., could be considered when generating an appropriate allocation.

C. Control Dissemination

Under different modality allocations, nodes might behave differently when making routing decisions. Thus, we should try to avoid the incidents such as packet loss, retransmission and looping etc., caused by inconsistent local forwarding behaviors. Pangu uses *version number* (VN), a common mechanism in data dissemination protocols [22][23], to identify which allocation a node is now referring to.

After generating a feasible allocation, Pangu controller delivers it with a new VN to the sink via the serial communication. Then, the sink disseminates them to the whole network rather than to one or some specific nodes at a time, because the ORW protocol is designed for data collection tasks and it would take too much effort to implement peer-to-peer communication.

It takes time to flood a packet to the entire network over multi-hop wireless links, especially for duty-cycled WSNs. Thus, during the dissemination of a new allocation, there could be a period when part of the nodes get the latest allocation while the others do not. The nodes are not required to switch to the latest allocation synchronously, because the time synchronization would incur an additional cost. Instead, a node instantly updates its local allocation upon receiving an updated one. However, this light-weighted design might incur locally-inconsistent forwarding behaviors due to the differences between 2 contiguous allocations.

Besides data packets $Data$, extra control packets $Request$ and $Inform$ are brought in to assist the control dissemination. A node broadcasts $Request$ packets to ask its neighbors for the latest allocation. $Inform$ packets containing a modality allocation are either to passively answer a $Request$ or to proactively flood the latest allocation. Every packet is marked with the VN of the node who sends it, in order to make local inconsistencies of VNs easier to spot.

Proactive broadcast. Next, we will introduce the two-phase procedure of the control dissemination. The first phase is called *proactive broadcast*, during which a node v proactively broadcasts a newly-arrived $Inform$, if $\mathrm{MF}(v) \neq \varnothing\}$. We assume that $\{v \in V|\mathrm{MF}(v) \neq \varnothing\}$ should construct a connected dominating set of the network. However, although the proactive broadcast lasts one working period, v's neighbors might miss all these proactive $Inform$s due to the channel interference or conflicts. The eventual consistency of VNs is

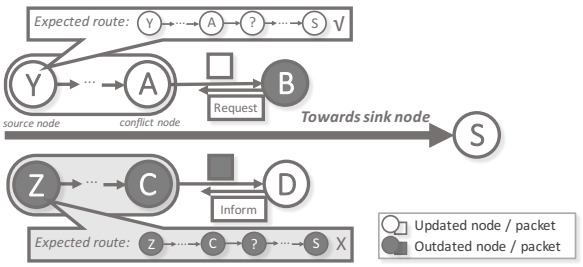


Fig. 4. **Two categories of inconsistencies during control dissemination.**

guaranteed by the second phase *passive coordination*, during which the latest allocation spreads across the entire network by exploiting local VN inconsistencies.

Passive coordination. First, we categorize the incidents caused by local VN inconsistencies. As shown in Figure 4, we can categorize the inconsistent cases into 2 types: Along the direction towards the sink,

1) Case 1: An updated node Y generates a packet p_1. Along one of the expected routes towards sink, an updated node A forwards p_1. Then, it happens to be heard by an outdated node B, who doesn't know how to handle p_1 under the latest allocation.
2) Case 2: An outdated node Z generates a packet p_2. Along one of the expected routes towards sink, an outdated node C forwards p_2. Then, it happens to be heard by an updated node D, who only knows how to handle p_2 under the latest allocation.

The source node (e.g. Y), the conflict node (e.g. A) and all the other nodes between them along the expected route must have the same VN, otherwise the conflict occurs before. Thus, in Case 1, if we can update all subsequent nodes of A (e.g. B) along this route to the latest version, p_1 from Y can still reach sink node along this expected route. However, some packets already buffered in B before B gets updated might not accord with new modal filters of B and B's descendants and may be discarded by them. In Case 2, the expected route built under some outdated allocation may be torn apart under the latest one. Updated subsequent nodes may no longer forward packets of the same modality as p_2. Moreover, the worst case is that there isn't any intersection between former and latter expected routes, in which case p_2 must be returned to its source Z.

The similar cases of inconsistent network behaviors do happen in SDN for wired networks and the predefined *default behavior* is often used to avoid packet loss or looping [24]. Pangu inherits similar ideas and leverages the fact that data packets always flow towards the sink to solve the above problems. We introduce *omni-modality*, which can pass any non-empty modal filter, to play the role of the default behavior. Packets marked with omni-modality can be forwarded by any potential nodes who have the routing progress, and finally will be delivered to the sink. Detailed rules of the passive coordination are as follows:

- When an outdated node B receives a packet p_1 from an updated node A, B first marks all packets in the forwarding queue with omni-modality, discards p_1 and does not reply an ACK. Then, B continuously broadcasts a $Request$ for the latest allocation and suspends its normal work until getting updated.
- When an updated node D receives a packet p_2 from an outdated node C, D first marks p_2 with omni-modality, buffers it and replies an ACK. Then D broadcasts an $Inform$ for a working period to make C and all the other outdated neighbors aware of the latest allocation.

V. Implementation and Evaluation

A. System Implementation

Pangu architecture contains two parts of components, the protocol stack and the controller. We implement the protocol stack by integrating our approaches into the original ORW stack on TinyOS. We keep the basic modules that maintain the neighbor table and compute EDC distributedly, and implement aforementioned centrally-controlled forwarding behaviors, state report mechanisms and control dissemination strategies. In our prototype system, NM is represented as a 8-bit vector and thus there are 2^8 types of forwarding behaviors among the nodes. The header of a Pangu packet only takes additional 4 bytes. The controller, written in Java 7, fulfills aforementioned components in §III.

B. Testbed-Based Evaluation

we conduct the evaluation of Pangu on a test-bed of 50 commercial TelosB Motes. The test-bed is a 5×10 grid and the distance between any two adjacent motes is 20cm. We put a unique sink at a corner of the test-bed, and connect it to the controller running on an Ubuntu system through the USB serial port. With the transmission power set to a minimum of -25dBm, the network diameter is about 4 to 5 hops under the above deployment. In the following experiments, the LPL period of the motes is set to 2000ms. Basically, the data sampling period is 30s and the beaconing period is 2min.

The evaluation contains 3 parts, an observation of network fluctuation during dense control disseminations, an analysis of allocation convergence time under different circumstances and a measurement of system performance with multi-function tasks. All of the modality allocations in these experiments are automatically generated by the minimum-radius generation algorithm (§IV-B).

C. Impact of Control Dissemination on Duty Cycle

When a new allocation is disseminated throughout the network with the help of the proactive broadcast and the passive coordination, the system performance is likely to decrease for a short period. In this experiment, we analyze the impact of the control dissemination on the system performance by observing the duty cycle fluctuation among the nodes.

We set the sampling period to 120s and trigger 4 continuous control disseminations with an average interval of 100s. The dissemination interval is comparable to the sampling period, thus we can observe and compare the fluctuations of the performance caused by these two factors. In Figure 5, we plot the duty cycle fluctuations of the overall moving average, the

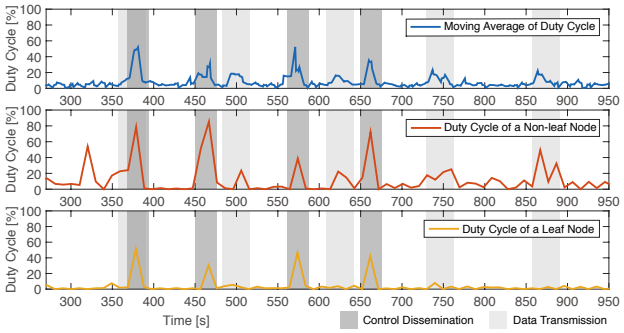


Fig. 5. **An observation of duty cycle fluctuation during the control dissemination.**

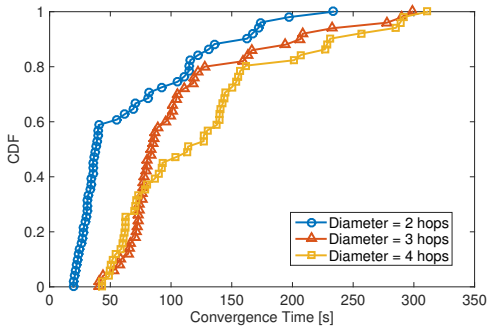


Fig. 6. **The CDF of convergence time under 3 different scales.**

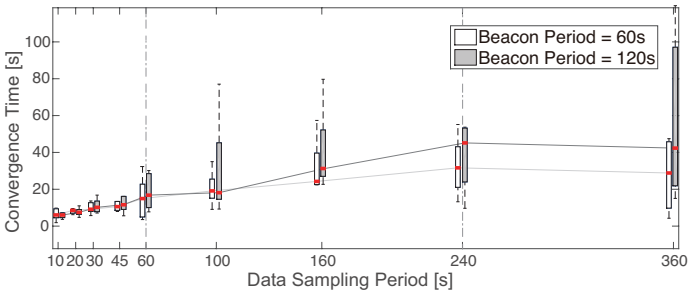


Fig. 7. **Convergence time bounded by beaconing period and data sampling period.**

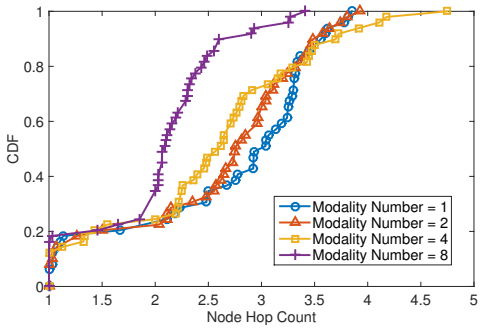


Fig. 8. **CDF of node hope count under different numbers of modalities.**

duty cycle of a non-leaf node and the duty cycle of a leaf node.

We can tell from the moving average of duty cycle that Pangu achieves a low average duty cycle among nodes in the stable network, and the control dissemination increases the average duty cycle of the nodes but this impact is acceptable compared to normal data transmission. By comparing the duty cycle of the nodes with different forwarding behaviors, we verify that a non-leaf node has a relatively higher duty cycle because it does more work during the proactive broadcast and the passive coordination.

D. Convergence Time under different circumstances

1) Convergence Time and Scale: The interval between the time when the sink starts disseminating an allocation and the time when the last node in the network gets this allocation is regarded as the convergence time. Therefore, the expected upper bound of the convergence time is in direct proportion to network diameter.

In this experiment, we measure the convergence times of three grid networks of different scales (4×4, 5×7 and 5×10). For each scale, we sample the convergence time for 50 times and plot the CDF in Figure 6. From the CDF, we could tell that the network with a larger diameter has a longer average convergence time, and in a specific network, the system converges quickly in most cases. However, in some cases, it still takes a relatively long time (e.g. 300s, 2.5X the data sampling period) for the network to converge, since our system can ensure the reliability of packet forwarding without the requirement for the strong consistency of the allocation version.

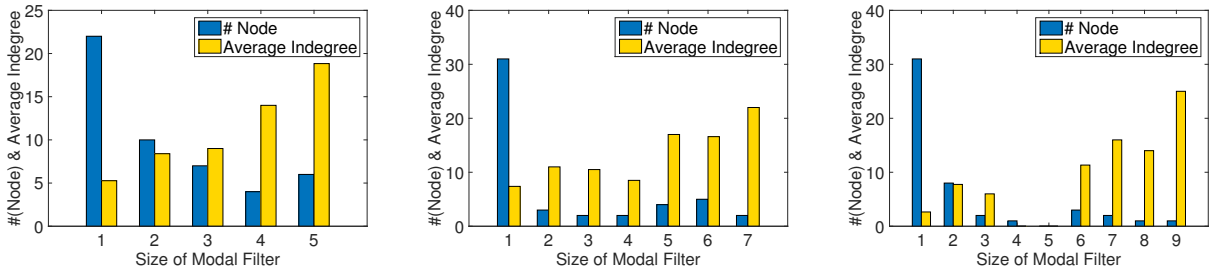
2) Convergence Time and Transmission Period: Other two factors, the beacon period T_b and the data sampling period T_{ds} might bound the convergence time T due to the strategies of the passive coordination. In this section, we compare convergence times under the parameters of $T_b \in \{T_{b1} = 60s, T_{b2} = 120s\}$ and T_{ds} ranging from $10s$ to $360s$. For each pair of T_b and T_{ds}, we measure T for 20 times.

Figure 7 shows the box plot of T under different configurations. We can conclude from the results that when $T_{ds} \leq T_{b1}$, T_{ds} bounds T, because regardless of the value of T_b, T increases slightly as T_{ds} increases. And when $T_{b1} < T_{ds} \leq T_{b2}$, either T_b or T_{ds} can bound T, because a decrease of T_b or T_{ds} could lead to a reduction in T.

E. Multi-Function Pangu Evaluation

Pangu enables multi-function WSNs by providing multiple choices of node modalities for the nodes. In this experiment, we evaluate Pangu under different settings of the number of modalities M, and find that a proper choice of M can benefit the performance of Pangu in a dense WSN. With M ranging from 1 to 8, we randomly assign NMs, and measure the average hop count for each node over a $30min$ stable time.

As shown in Figure 8, we find that the average hop count diminishes while M increases. As M increases, the randomly-generated modality collection becomes more discrete and a node may have fewer but better choices of forwarders. Our modality allocation algorithm always provides a node with multiple optimal forwarders in a simplified model, thus if more choices are offered, the packets from this node have a higher probability to be relayed by those forwarders that are actually sub-optimal.



(a) M = 4 (b) M = 6 (c) M = 8

Fig. 9. **Correlation between the in-degree of a node and the size of its modal filter under different values of M.**

Moreover, we analyze the correlation between the in-degree of a node and the size of its modal filter, which represents the number of modalities it is allowed to pass, in order to evaluate the effectiveness of our generation algorithm. Figure 9 shows that basically fewer nodes have a larger size of modal filter, regardless of the number of modalities M. The result suggests that the actual work these nodes does can match their centrally-defined forwarding behaviors, which means our modality-based software-defined approach and corresponding allocation generation algorithm can effectively present the underlying routing structure.

VI. Conclusion

In this paper, we propose Pangu, an innovative software-defined architecture for multi-function wireless sensor networks. Pangu enables network users to centrally define the forwarding behaviors of nodes in the underlying opportunistic routing protocol. This feature is obtained by introducing the concept of modality properties of the sensor nodes. With the help of centrally-defined modality properties, the network is partitioned into multiple desirable sub-nets that can satisfy the upper-level requirements for system performance. Moreover, we explore the critical problems of the architecture design, such as the maintenance of global states, the allocation and dissemination of modality properties. Finally, we evaluate the performance of a prototype system, and the corresponding results demonstrate the feasibility and the effectiveness of Pangu.

Acknowledgment

This work is supported in part by National Natural Science Foundation of China No. 61672320 and No. 61772306, National Natural Science Fund for Excellent Young Scientist No. 61422207, National Basic Research Program (973) No.2014CB347800 and China Postdoctoral Science Foundation No. 2016M601034.

References

[1] J. Yick, B. Mukherjee, and D. Ghosal, "Wireless sensor network survey," *Computer networks*, 2008.

[2] P. Rawat, K. D. Singh, H. Chaouchi *et al.*, "Wireless sensor networks: a survey on recent developments and potential synergies," *The Journal of supercomputing*, 2014.

[3] S. Jain, A. Kumar, S. Mandal *et al.*, "B4: Experience with a globally-deployed software defined wan," in *Proceedings of ACM SIGCOMM*, 2013.

[4] M. Alizadeh, T. Edsall, S. Dharmapurikar *et al.*, "Conga: Distributed congestion-aware load balancing for datacenters," in *Proceedings of ACM SIGCOMM*, 2014.

[5] N. McKeown, T. Anderson, H. Balakrishnan *et al.*, "Openflow: enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, 2008.

[6] S. Biswas and R. Morris, "Exor: opportunistic multi-hop routing for wireless networks," in *Proceedings of ACM SIGCOMM*, 2005.

[7] O. Landsiedel, E. Ghadimi, S. Duquennoy, and M. Johansson, "Low power, low delay: opportunistic routing meets duty cycling," in *Proceedings of ACM/IEEE IPSN*, 2012.

[8] D. S. De Couto, D. Aguayo, J. Bicket, and R. Morris, "A high-throughput path metric for multi-hop wireless routing," *Wireless Networks*, 2005.

[9] P. Levis, N. Patel, D. Culler, and S. Shenker, "Trickle: A self-regulating algorithm for code propagation and maintenance in wireless sensor networks," in *Proceedings of USENIX/ACM NSDI*, 2004.

[10] R. K. Panta and S. Bagchi, "Hermes: Fast and energy efficient incremental code updates for wireless sensor networks," in *Proceedings of IEEE INFOCOM*, 2009.

[11] C. C. Han, R. Kumar, R. Shea *et al.*, "A dynamic operating system for sensor nodes." in *Proceedings of ACM MobiSys*, 2005.

[12] C. Cao, L. Luo, Y. Gao, and W. Dong, "Tinysdm: software defined measurement in wireless sensor networks," in *Proceedings of ACM/IEEE IPSN*, 2016.

[13] M. Kochhal, L. Schwiebert, and S. Gupta, "Role-based hierarchical self organization for wireless ad hoc sensor networks," in *Proceedings of ACM WSNA*, 2003.

[14] C. Frank and K. Römer, "Algorithms for generic role assignment in wireless sensor networks," in *Proceedings of ACM Sensys*. ACM, 2005.

[15] T. Miyazaki, S. Yamaguchi, K. Kobayashi *et al.*, "A software defined wireless sensor network," in *Proceedings of IEEE ICNC*, 2014.

[16] T. Luo, H.-P. Tan, and T. Q. Quek, "Sensor openflow: Enabling software-defined wireless sensor networks," *Communications Letters, IEEE*, 2012.

[17] D. Zeng, P. Li, S. Guo *et al.*, "Energy minimization in multi-task software-defined sensor networks," *IEEE Transactions on Computers*, 2015.

[18] L. Galluccio, S. Milardo, G. Morabito, and S. Palazzo, "Sdn-wise: Design, prototyping and experimentation of a stateful sdn solution for wireless sensor networks," in *Proceedings of IEEE INFOCOM*, 2015.

[19] S. Vissicchio, O. Tilmans, L. Vanbever, and J. Rexford, "Central control over distributed routing," in *Proceedings of ACM SIGCOMM*, 2015.

[20] A. Ghosh, O. D. Incel, V. S. A. Kumar, and B. Krishnamachari, "Multichannel scheduling and spanning trees: throughput-delay tradeoff for fast data collection in sensor networks," *IEEE Transactions on Networking*, 2011.

[21] M. K. An, N. X. Lam, D. T. Huynh, and T. N. Nguyen, "Bounded-degree minimum-radius spanning trees in wireless sensor networks," *Theoretical Computer Science*, 2013.

[22] X. Zheng, J. Wang, W. Dong *et al.*, "Bulk data dissemination in wireless sensor networks: analysis, implications and improvement," *IEEE Transactions on Computers*, 2016.

[23] T. Dang, N. Bulusu, W.-C. Feng, and S. Park, "Dhv: A code consistency maintenance protocol for multi-hop wireless sensor networks," in *Wireless sensor networks*, 2009.

[24] A. Gupta, L. Vanbever, M. Shahbaz *et al.*, "Sdx: a software defined internet exchange," in *Proceedings of ACM SIGCOMM*, 2015.