软件定义 WSN 规则一致更新研究

黄美根,郁 滨
(信息工程大学,河南郑州 450001)

摘 要: 遵循控制转发分离思想,软件定义无线传感器网络(Wireless Sensor Network,WSN)数据转发采用基于流的实现方式. 因此,软件定义 WSN 规则更新过程中节点行为可能违背网络属性一致性. 针对此,提出每包前向一致性概念,并证明其可保持所有网络属性的更新一致性. 在此基础上,通过引入缓存节点与缓存规则简化规则依赖关系,提出一种规则前向一致更新算法,在满足每包前向一致性的同时,支持规则快速并行更新. 实验结果表明,算法在规则开销、更新时间和通信开销等关键性能指标上具有较为明显的优势.

关键词: 无线传感器网络; 软件定义网络; 物联网; 软件定义无线传感器网络; 网络安全; 规则更新; 每包前向一致性

中图分类号: TP393 **文献标识码:** A **文章编号:** 0372-2112 (2019)09-1965-07

电子学报 URL: http://www.ejournal.org.cn **DOI:** 10.3969/j.issn.0372-2112.2019.09.021

Research on Consistent Rule Update in Software-Defined WSN

HUANG Mei-gen, YU Bin
(*Information Engineering University, Zhengzhou, Henan* 450001, *China*)

Abstract: Following the idea of separation of control and forwarding, the data forwarding of software-defined WSN (Wireless Sensor Network) is implemented in a flow-based manner. Therefore, nodes behavior during rule updating in software-defined WSN may violate network attribute consistency. Therefore, this paper proposes the concept of per-package forwarding consistency and proves that it can maintain the update consistency of all attributes. On this basis, a rule forwarding consistent update algorithm is proposed by introducing cache nodes and rules to simplify dependencies. The algorithm supports fast parallel updates while satisfying the per-package forwarding consistency. The experimental results show that the algorithm has obvious advantages in the rule cost, update time and communication overhead.

Key words: wireless sensor network; software-defined networking; Internet of Things (IoT); software-defined wireless sensor network; network security; rule update; per-package forwarding consistency

1 引言

随着物联网(Interne of Things,IoT)的蓬勃发展,作为其感知层关键支撑技术的传统无线传感器网络(Wireless Sensor Network,WSN)[1]在资源互连共享、环境动态感知等方面面临的挑战日益严峻[2,3],采用软件定义网络(Software-Defined Networking,SDN)[4]思想的新型 WSN 逐渐获得青睐,即软件定义 WSN(Software-Defined WSN,SDWSN)[5,6]. 通常,SDWSN 遵循 SDN 控制转发分离原则,流表规则(简称规则)由控制面生成,数据面仅通过匹配规则实现网络流(一系列具有某些相同性质的包)的转发[7].

由于无线通信的动态性和用户需求的多样性(SDWSN 支持面向用户提供网络编程服务),控制面需要对数据面规则进行快速更新,以保证对数据包的正确处理. 然而,传输时延的不确定性使得实际网络中节点规则更新时间和数据包到达时间都难以预测[8]. 因此,规则更新过程中网络可能处于新旧规则混用的状态,从而使节点转发行为破坏网络属性的一致性,如转发环路、路由黑洞等[9],甚至违反访问控制策略,对网络安全构成严重威胁[10]. 缘于 SDWSN 尚处于发展初期,目前学术界并未诞生与 SDWSN 规则一致更新相关的研究成果. 但是,针对 SDN,学者已提出大量解决方案. 考虑到传感节点资源受限特性,这些方案并不能直接适

收稿日期:2018-10-31;修回日期:2019-04-24;责任编辑:覃怀银
基金项目:国家自然科学基金(No.61602513);信息保障技术重点实验室开放基金(No.KJ-15-104)

用于 SDWSN,但仍然具有一定借鉴价值.

面向 SDN,Reitblatt 等[11]提出了每包一致性概念,即每个数据包在整个转发过程中仅匹配新规则或旧规则,并据此思路提出了两阶段更新方案,通过在交换机上同时安装新旧两套规则分别匹配具有新旧标签的数据包,从而实现规则一致更新. 尽管该方案具有较好的通用性和可操作性,但所需规则开销过大,极大制约了实用性. 于是,Katta 等[12]采取以更新时间换取规则开销的思路,提出增量一致更新方案,将两阶段更新划分为 k 轮子更新,每轮在满足一致性的前提下更新部分规则. 虽然方案通过 6 轮更新即可将规则开销从 100% 降为 10%,但确定最优子更新需求解混合整数线性规划,进一步增加了更新时间. 无独有偶,Mcgeer 等[13]采取以控制器负载换取规则开销,提出一种安全更新协议,通过构建中间规则将所有可能破坏网络属性一致性的数据包均发送至控制器进行缓存,待规则更新完成后重新注入网络. 尽管该协议可以保证网关在任何时候仅需安装一套规则,但其在加剧控制器负担的同时,引入了过多通信开销[14].

综上所述,满足每包一致性的更新方案存在规则开销较大、更新速度较慢和通信开销较高等问题,根本原因在于每包一致性要求较为严格,使得规则之间依赖关系过于复杂[15]. 因此,本文提出每包前向一致性概念,并在采用缓存节点和缓存规则简化依赖关系的基础上,设计满足网络属性更新一致性的规则前向一致更新(Rule Forwarding Consistent Update,RFCU)算法.

2 每包前向一致性

在每包一致性概念的基础上,通过允许数据包至多一次从匹配旧规则正向迁移至匹配新规则而不允许反向迁移,给出每包前向一致性概念如定义 1 所示.

定义 1 每包前向一致性指网络规则更新过程中数据包一旦匹配新规则便不再匹配旧规则.

性质 1 网络属性更新一致性,即保持规则更新过程中网络不出现转发环路、路由黑洞等不期望属性,记为 cons.

为便于阐述,在定义前向节点及转发函数的基础上,给出每包前向一致性的数学表示(定理 1),然后据此证明每包前向一致性满足性质 1(定理 2). 记 p 为数据包,运算符“$\succ$”代表“满足”操作.

定义 2 节点 $s_{q'}$ 是节点 s_q 的前向节点指 p 的转发必须先经过 s_q 后才能到达 $s_{q'}$,记为 $s_q \mapsto s_{q'}$,“$\mapsto$”为前向符.

定义 3 节点转发数据包的映射称为节点转发函数,记为 $f_u^s: X \to Y$,其中 X 为规则匹配域,Y 为规则动作域,$u \in \{1,2\}$ 表示转发所匹配的规则集(“1”表示旧规则,“2”表示新规则).

定义 4 网络转发数据包的映射称为网络转发函数,记为 $h_v: X \to Y$,其中 X 为入口节点规则匹配域,Y 为出口节点规则动作域,v 为网络映射规则,表示 p 在网络中转发所匹配规则序列的标识.

由定义 3 和定义 4 可知,网络转发函数可以表征为节点转发函数序列,即 $h_v(p) = f_{u_r}^{s_r}(\cdots(f_{u_2}^{s_2}(f_{u_1}^{s_1}(p)))\cdots)$,其中 $u_j \in u, j \in \{1,2,\cdots,r\}$,$p$ 的转发路径为 $(s_1, s_2, \cdots, s_r)$,s_1 和 s_r 分别为入口和出口节点.

依据 u_j 的取值,给出 v 取值如式(1)所示.

$$v = \begin{cases} 1, & u_j \equiv 1 \\ 2, & u_j \equiv 2 \\ 12, & \forall u_j \le u_{j+1} \cap \exists u_j < u_{j+1} \\ 21, & \exists u_j > u_{j+1} \end{cases} \tag{1}$$

因此,$h_1(p)$、$h_2(p)$、$h_{12}(p)$、$h_{21}(p)$ 分别表示 p 在 $(s_1, s_2, \cdots, s_r)$ 中仅匹配旧规则、仅匹配新规则、在 s_j 前匹配旧规则而 s_{j+1} 后匹配新规则、在 s_j 处匹配新规则后在其前向节点 s_{j+1} 处匹配旧规则. 此外,记每包前向一致性为 $h(p)$.

引理 1 $(s_q \mapsto s_{q'}) \cap f_2^s(p) \Rightarrow f_2^{s'}(p)$.

证明 由定义 1 可知 p 匹配新规则后不再允许匹配旧规则,因此 p 在 s_q 处匹配新规则后,在其前向节点 $s_{q'}$ 处必然匹配新规则,即 $(s_q \mapsto s_{q'}) \cap f_2^s(p) \Rightarrow f_2^{s'}(p)$. 证毕.

定理 1 $h(p) = h_1(p) \cup h_2(p) \cup h_{12}(p)$.

证明 由定义 4 可知,$h(p) = h_1(p) \cup h_2(p) \cup h_{12}(p) \cup h_{21}(p)$,且满足 $h_{v_i}(p) \cap h_{v_j}(p) = \varnothing$,$v_i \in v, v_j \in v, v_i \neq v_j$. 因此 $h(p) = h_1(p) \cup h_2(p) \cup h_{12}(p) \Leftrightarrow h_{21}(p) \not\subset h(p)$.

采用反证法,反命题为 $\exists p \Rightarrow h(p) = h_{21}(p)$. 显然,由定义 3 和定义 4 可知 $h_{21}(p) = \cdots(f_1^{s_{j+1}}(f_2^{s_j}(p)))\cdots$,其中 $s_j \mapsto s_{j+1}$. 由引理 1 可知 $(s_j \mapsto s_{j+1}) \cap f_2^{s_j}(p) \Rightarrow f_2^{s_{j+1}}(p)$,不存在 p 满足 $h_{21}(p)$,即 $\forall p \Rightarrow h(p) \neq h_{21}(p)$. 因此反命题不成立,$h_{21}(p) \not\subset h(p)$ 得证,即 $h(p) = h_1(p) \cup h_2(p) \cup h_{12}(p)$. 证毕.

引理 2 $h_1(p) \cup h_2(p) \succ \text{cons}$.

证明 $h_1(p) \cup h_2(p)$ 为每包一致性的数学表示,满足性质 1,证明过程详见文献[11]. 证毕.

定理 2 $h(p) \succ \text{cons}$.

证明 由定理 1 可得 $h(p) \succ \text{cons} \Leftrightarrow (h_1(p) \succ \text{cons}) \cup (h_2(p) \succ \text{cons}) \cup (h_{12}(p) \succ \text{cons})$. 由引理 2 可得 $h_1(p) \succ \text{cons}$ 与 $h_2(p) \succ \text{cons}$ 成立,因而 $h(p) \succ \text{cons} \Leftrightarrow h_{12}(p) \succ \text{cons}$.

由定义 3 和定义 4 可知,$h_{12}(p) = f_2^{s_r}(\cdots(f_2^{s_{j+1}}(f_1^{s_j}(\cdots(f_1^{s_1}(p))\cdots)))\cdots)$,依据新旧规则匹配情况可将 $h_{12}(p)$

分解为两个子函数,即 $h_{12}(p)=h'_2(h'_1(p))$,其中 $h'_1(p)=f_1^{s_j}(\cdots(f_1^{s_1}(p))\cdots)$,$h'_2(p)=f_2^{s_j}(\cdots(f_2^{s_{j+1}}(p))\cdots)$. 显然,$h'_1(p)\subset h_1(p)\succ$cons,$h'_2(p)\subset h_2(p)\succ$cons. 因此,$h_{12}(p)\succ$cons$\Leftrightarrow h'_1(p)\cap h'_2(p)=\varnothing$.

令rule_1 和rule_2 分别表示节点 s_j 和 s_{j+1}之间的旧规则和新规则,$\text{rule}_1\otimes\text{rule}_2$ 表示冲突规则,"$\otimes$"为规则冲突符. 据此可知,当且仅当 p 分别匹配了rule_1 和rule_2 中的冲突规则,即$\text{rule}_1\cap\text{rule}_2$ 至少被 p 匹配 2 次,网络属性的一致性才可能被违背,如形成转发环路. 由rule_1 和rule_2 的关系可知 $h'_1(p)\cap h'_2(p)=\varnothing$仅在两种情况下成立:①$\text{rule}_1\otimes\text{rule}_2=\varnothing$;②$\text{rule}_1\otimes\text{rule}_2\neq\varnothing$且$\text{rule}_1\otimes\text{rule}_2$ 仅被 p 匹配 1 次. 针对①,显然有 $h'_1(p)\cap h'_2(p)=\varnothing$. 针对②,在匹配$\text{rule}_1$ 时有 $s_j\mapsto s_{j+1}$,而在匹配rule_2时则有 $s_{j+1}\mapsto s_j$. 由引理 1 可得$(s_{j+1}\mapsto s_j)\cap f_2^{s_{j+1}}(p)\Rightarrow f_2^{s_j}(p)$,因此 p 在节点 s_j 和 s_{j+1}之间的转发过程可以表示为$(f_1^{s_j}(p),f_2^{s_{j+1}}(p),f_2^{s_j}(p))$. 显然,$f_1^{s_j}(p)\otimes f_2^{s_j}(p)=\varnothing$,同时由新规则自身不冲突可知 $f_2^{s_{j+1}}(p)\otimes f_2^{s_j}(p)=\varnothing$,即$\text{rule}_1\otimes\text{rule}_2$ 仅在$(f_1^{s_j}(p),f_2^{s_{j+1}}(p))$中被 p 匹配 1 次. 因此,情况②下 $h'_1(p)\cap h'_2(p)=\varnothing$也成立. 综上,$h'_1(p)\cap h'_2(p)=\varnothing$成立,从而原命题 $h(p)\succ$cons 得证,即每包前向一致性满足性质 1. 证毕.

3　RFCU 算法

本节首先给出缓存节点集构造(Cache node Set Constructing,CSC)算法,并在此基础上设计缓存规则,最后给出满足每包前向一致性的 RFCU 算法.

3.1　缓存节点

定义 5　规则简化分界节点称为缓存节点,

定义 6　两个相邻缓存节点间规则称为子规则,相应更新区域称为更新域.

缓存节点控制面选择,包括合并规则缓存节点和超长规则缓存节点,分别表示规则路径中采用合并规则转发的节点以及超长子规则的分界节点. 给出 CSC 算法如算法 1 所示. 输入参数中,rule 为待简化规则,ss 为与 rule 相关的节点集,maxlength 为最大允许规则长度;输出参数 cs 为缓存节点集.

算法 1　CSC 算法

输入:rule;ss;maxlength;
输出:cs.
1:foreach $\{s\mid s\in ss\}$
2:　if $s_{\text{rule}}\in$ mergerule //mergerule 表示合并规则
3:　　$s\rightarrow$cs;　//将合并规则缓存节点 s 加入 cs
4:　　$\text{rule}_s=\text{rule}_{s-}\cup\text{rule}_{s+}$;
　　//将规则rule_s 简化为子规则rule_{s-}和rule_{s+}
5:　　end if
6:end for
7:foreach $\{\text{rule}_i\mid\text{rule}_i\in\text{rule}\}$
8:　if length(rule_i) > maxlength //验证子规则长度
9:　　num = length(rule_i)/maxlength;
　　//计算规则简化次数 num
10:　　$\text{rule}_i=\text{rule}_{i_1}\cup\text{rule}_{i_2}\cup\cdots\cup\text{rule}_{i_{\text{num}+1}}$;
　　//将rule_i 平均简化为 num + 1 条子规则
11:　　foreach $\{j\mid j\leq$num + 1$\}$
12:　　$i_j\rightarrow$cs;　//将超长规则缓存节点 i_j 加入 cs
13:　　end for
14:　end if
15:end for

3.2　缓存规则

定义 7　穿越不同更新域传输的数据包称为跨域数据包.

定义 8　规则更新过程中缓存节点匹配跨域数据包的规则称为缓存规则.

规则更新过程中,更新域内可能同时存在新旧规则,缓存节点转发跨域数据包可能违背每包前向一致性. 因此,设计缓存规则临时存储跨域数据包,保证各更新域的独立性. 基于节点转发函数和 CSC 算法,给出缓存规则 $\tilde{u}$ 设计如式(2)所示.

$$\begin{aligned}\tilde{u}_s&:\{X_{s_{\text{rule}}}\}\rightarrow\{\text{action}=\text{cache}\}\\ \tilde{u}_{i_j}&:\{X_{i_j}\}\rightarrow\{\text{action}=\text{cache}\}\end{aligned}\tag{2}$$

其中,$X_{s_{\text{rule}}}$为合并规则缓存节点 s 中隶属于 rule 的规则匹配域,X_{i_j}为超长规则缓存节点 i_j 的规则匹配域. 为平衡更新域独立性与规则开销,s 新增具有最高优先级的 $\tilde{u}_s$,而 i_j 则在原规则基础上修改动作域实现 $\tilde{u}_{i_j}$. 此外,动作 cache 表示存储,因此缓存节点需要维护一个存储队列用于存储跨域数据包. 当存储队列空间不够时,跨域数据包将发送至控制器进行存储.

3.3　算法描述

RFCU 算法采取分域并行更新思路. 通过将依赖关系复杂的规则简化为若干条依赖关系简单的子规则,规则更新空间被划分为若干更新域. 进一步,借助缓存规则,由缓存节点存储跨域数据包,实现更新域的独立性,从而保证并行更新满足每包前向一致性. 给出 RFCU 算法如算法 2 所示,rules 表示待更新的旧规则集,updatedrules 表示更新完成的新规则集,maxtime 为最大域内时延.

算法 2　RFCU 算法

输入:rules;cs;ss;maxtime;maxlength;
输出:updatedrules.
1:foreach $\{\text{rule}\mid\text{rule}\in\text{rules}\}$
2:cs = CSC(rule,ss,maxlength);　//构造缓存节点集
3:　foreach $\{c\mid c\in\text{cs}\}$

```
4:      if c_rule ∈ mergerule
5:          split c_rule:{X_1}→{Y_1} from mergerule;
                          //从合并规则中分离出 c_rule
6:          insert ũ_c:{X_1}→{action = cache};
                          //新增最高优先级缓存规则 ũ_c
7:      else c_rule ∉ mergerule
8:          read c_rule:{X'_1}→{Y'_1};  //从规则中读取 c_rule
9:          modify c_rule as ũ_c:{X'_1}→{action = cache};
                          //修改 c_rule 实现缓存规则 ũ_c
10:     end if
11:  end for
12:end for
13:sleep(maxtime);//等待最大域内时延,排空域内数据包
14:rules --parallelupdating--> updatedrules;
                          //并行更新 rules 至 updatedrules
15:delete f_ũ^c;  //域内更新完后删除缓存规则
```

算法2中,第2行表示通过调用CSC算法构造缓存节点集cs;第4~6行负责处理cs中所有合并规则缓存节点,采取先分离原规则后增加缓存规划的方式实现;第7~9行则针对剩余所有非合并规则的缓存节点(即为超长规则缓存节点),在读取原规则的基础上修改实现. 在此基础上,第13行采取等待最大域内时延的方式排空域内数据包,随后以并行下发更新方式从旧规则集rules切换至新规则集updatedrules(第14行),排空域内数据包为防止其在并行更新过程中违背网络属性更新一致性. 最后,在并行更新完成后,删除所有缓存规则(第15行),网络以新规则集状态运行.

4 性能分析

表1为本文与相关文献的性能定性对比分析结果,包括规则开销、通信开销、存储开销、计算开销以及更新速度等五个性能指标. 其中,文献[11~13]分别为适用于SDN的两阶段更新方案、增量一致更新方案、安全更新协议,L_1、L_2、L_3分别表示低、中、高,S_1、S_2、S_3分别表示快、中、慢.

表1 性能对比分析

性能指标	文献[11]	文献[12]	文献[13]	RFCU
规则开销	L_3	L_2	L_1	L_1
通信开销	L_1	L_1	L_3	$L_1{}^*$
存储开销	L_1	L_1	L_2	L_2
计算开销	L_1	L_3	L_1	L_1
更新速度	S_1	S_3	S_2	$S_1{}^*$

由表1可知,本文RFCU算法较好地平衡了规则开销、通信开销、存储开销以及计算开销等更新代价,且具有较快地更新速度. 其中,当RFCU算法中较多缓存节点存储队列满时,$L_1{}^*$可能上升至L_2,$S_1{}^*$可能降低至S_2.

(1)规则开销. RFCU算法仅在合并规则缓存节点处需增加一条缓存规则,其它节点均可保证任何时候仅安装一套规则,与安全更新协议基本相同(~0%);两阶段更新则采取同时安装新旧两套规则,因而需要两倍的规则空间(100%);而增量一致更新则在两阶段更新的基础上通过增加更新轮数来合理降低规则开销(0%~100%,6轮更新为10%). 有必要说明的是,为提升匹配速度,实际应用中流表转发通常采取硬件实现,因此降低规则开销可以有效缩减节点成本[16].

(2)通信开销. 除所有方案都必须传输的新规则外,两阶段更新通信开销最低,而本文通信开销主要包括缓存规则传输(缓存节点数量较少)和跨域数据包传输(近距离),总的通信开销仍然较低,与增量一致更新方案基本持平. 安全更新协议采取将所有数据包发送至控制器(远距离)进行存储,这与RFCU算法的极端情况(存储队列满时将跨域数据包发送至控制器存储)相同,必然引入较多通信成本.

(3)存储开销. 与安全更新协议相似,RFCU算法也采取存储可能违背性质1的数据包来保证规则更新一致性. 不同的是,安全更新协议采取全部发往控制器存储,而RFCU算法则由缓存节点就近存储,从而获得通信成本和更新时间上的优势,但根本上两者存储开销相同. 两阶段更新与增量一致更新均无需缓存数据包,因此存储开销较低.

(4)计算开销. 增量一致更新方案需要运行混合整数线性规划来求解最优子更新,需要消耗较多计算资源,而安全更新协议与RFCU算法均只需计算中间规则或缓存规则,计算开销较低,与两阶段更新方案持平.

(5)更新速度. 两阶段更新方案可保证1轮完成更新(速度较快),而增量一致更新则需要k轮子更新(速度较慢). 安全更新协议通过安装中间规则来节约规则开销,但其控制器存储方式在一定程度上降低了更新速度. RFCU算法通过简化规则依赖关系,在并行更新的同时就近存储跨域数据包,因而具有比安全更新协议更快的更新速度,略低于两阶段更新方案.

5 实验与结果分析

本文实验基于SDN-WISE[17]设计,其是一款基于OpenFlow的开源SDWSN架构,实验基于COOJA仿真平台实现. 依据表1,选择两阶段更新方案[11]、增量一致更新方案[12](6轮)和安全更新协议[13]为对比方案. 考虑到规则开销已在性能分析一节中给出了数值对比结果,因此本实验选择更新时间和通信开销作为主要测试指标.

5.1 实验部署

为便于评估规则更新性能,采取更换数据收集节

点模拟网络状态切换. 控制面部署单控制器,数据面以链状拓扑部署 30 个节点,如图 1 所示.

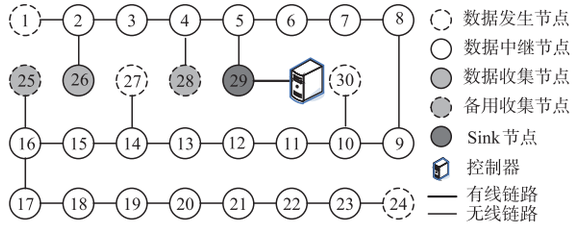

图1　实验拓扑结构图

图 1 中,部署数据发生节点 4 个(1 号、24 号、27 号和 30 号节点),数据包产生速率设定为 speed ∈ {0,10,20,30,40},单位为包每秒(packet per second,pps),包负载为 50 字节;2 ~ 23 号节点为数据中继节点;26 号节点为初始数据收集节点,25 号和 28 号节点为备用收集节点,用于模拟两种不同规模的规则更新场景,分别记为局部更新(26→28)和全局更新(26→25);29 号节点为 Sink 节点,用于连通控制面与数据面. 此外,设定缓存节点存储队列大小 cache ∈ {2KB,64KB},maxlength 和 maxtime 分别取值为 4 和 30ms.

5.2　实验结果

5.2.1　更新时间

更新时间指从控制面接收网络状态改变报告到网络完成状态切换的时间,实验中使用数据收集节点切换命令代替网络状态改变报告. 给出更新时间实验结果如图 2 所示.

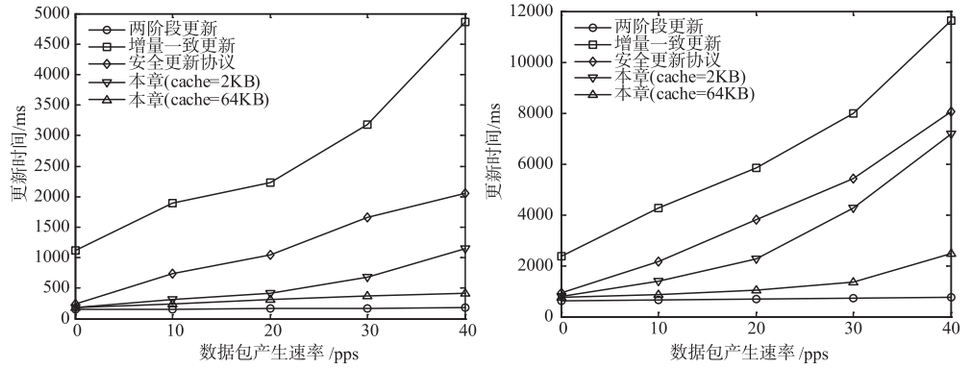

图2　更新时间实验结果

图 2 中,无论是局部更新还是全局更新,两阶段更新方案所需时间最少,增量一致更新方案所需时间最多,原因在于前者以 100% 的规则开销保证了数据包不会违背性质 1,而后者在降低规则开销的同时却引入了较长的多轮子更新以及规划求解时间.

与安全更新协议相比,本文 RFCU 算法在更新时间方面具有较大优势. 一方面,算法在更新域的基础上采取并行更新极大加速了更新进程. 另一方面,采取缓存节点就近存储进一步减少了跨域数据包的传输时间. 在局部更新和低速全局更新中,本文缓存节点基本可以满足需求,而在较为极端的情况下(全局更新中 speed≥30pps 且 cache = 2KB),更新时间增长较快,与安全更新协议相近. 给出全局更新(speed = 30pps)下所有方案包传输时间对比结果如图 3 所示. 其中,X 轴为以对数形式表征的包传输时间,Y 轴为累积分布函数 CDF.

包传输时间指数据包从源节点到目的节点之间传输所经历的时间,包括数据包无线传输时间、处理时间、排队时间以及缓存时间等. 由图 3 可知,两阶段更新和增量一致更新的包传输时间最短,介于 28.4ms 至

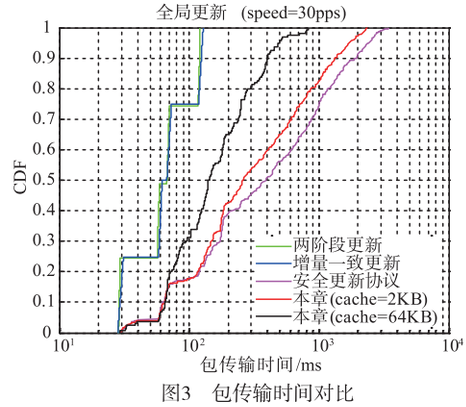

图3　包传输时间对比

128.2ms 之间,这也从侧面印证了多轮子更新及规划求解时间在总时间中比重较大. 当 cache = 64KB 时,RFCU 算法中大部分跨域数据包被就近存储于缓存节点;而当 cache = 2KB,缓存队列迅速饱和,跨域数据包最终被发往控制器存储,因此包传输时间增长趋势与安全更新协议相似.

5.2.2　通信开销

规则更新过程中节点产生数据包数量对通信开销

具有较大影响,给出通信开销实验结果如图 4 所示. 通信开销不包括沿新旧规则指定路径传输的数据包.

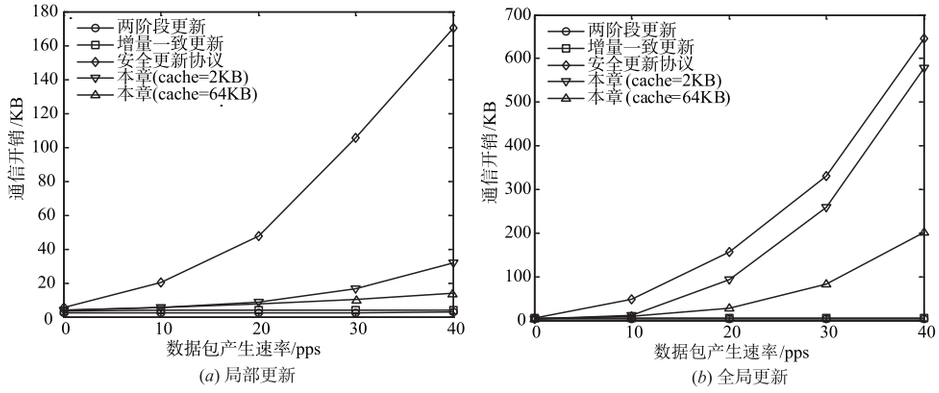

图4 通信开销实验结果

由图 4 可知,两阶段更新和增量一致更新方案通信开销极低,原因在于网络数据包不会影响两者规则更新的一致性,因而不存在缓存操作. 而安全更新协议采取将全部数据包发往控制器存储的方式必然引入过多通信代价,在局部更新和全局更新下通信开销最高分别达 170. 2KB 和 645. 5KB,且随着网络规模的增大呈指数增长.

RFCU 算法中,跨域数据包首先就近存储于缓存节点,当缓存队列满时才发往控制器存储,因而仅当网络中数据包过多时,通信开销才显著增加. 显然,算法 cache 取值越大,缓存跨域数据包越多,通信开销就越小,但这是以节点存储空间为代价的. 因此,节点 cache 取值需要依据网络实际情况而定. 需要说明的是,尽管两阶段更新方案兼具更新时间和通信开销上的显著优势,但 100% 的规则开销是制约其在实际网络中应用的关键.

6 总结

本文在深入分析 SDWSN 规则更新特点的基础上,提出了每包前向一致性概念,并证明了其可以保持更新过程中所有网络属性的一致性. 进一步,通过引入缓存节点和缓存规则在简化规则依赖关系的同时保证规则更新域的独立性,并基于此提出了满足每包前向一致性的 RFCU 算法. 最后,在对算法进行性能分析的基础上,设计实验对更新时间和通信开销进行验证,结果表明 RFCU 算法在规则开销、更新时间和通信开销等关键性能指标上具有较好的综合性能,可以很好地适用于资源受限的 SDWSN.

参考文献

[1] Borges L M, Velez F J, Lebres A S. Survey on the characterization and classification of wireless sensor network applications[J]. IEEE Communications Surveys & Tutorials, 2014, 16(4): 1860 – 1890.

[2] Huang M, Yu B, Li S. PUF-assisted group key distribution scheme for software-defined wireless sensor networks[J]. IEEE Communications Letters, 2018, 22(2): 404 – 407.

[3] Valdivieso Caraguay A L, Benito Peral A, Barona Lopez L I, et al. SDN: Evolution and opportunities in the development IoT applications[J]. International Journal of Distributed Sensor Networks, 2014, 10(5): 735142: 1 – 10.

[4] Mc Keown N. Software-defined networking[A]. Proceedings of IEEE International Conference on Computer Communications[C]. Rio de Janeiro, Brazil: IEEE, 2009. 30 – 32.

[5] Luo T, Tan H P, Quek T Q S. Sensor OpenFlow: Enabling software-defined wireless sensor networks[J]. IEEE Communications Letters, 2012, 16(11): 1896 – 1899.

[6] 黄美根,黄一才,郁滨,等. 软件定义无线传感器网络研究综述[J]. 软件学报,2018,29(9):2733 – 2752.
Huang Mei-gen, Huang Yi-cai, Yu Bin, et al. Software-defined wireless sensor networks: A research survey[J]. Journal of Software, 2018, 29(9): 2733 – 2752. (in Chinese)

[7] Mahmud A, Rahmani R. Exploitation of OpenFlow in wireless sensor networks[A]. Proceedings of 2011 International Conference on Computer Science and Network Technology[C]. Harbin, China: IEEE, 2011. 594 – 600.

[8] Misra S, Bera S, Achuthananda M P, et al. Situation-aware protocol switching in software-defined wireless sensor network systems[J]. IEEE Systems Journal, 2018, 12(3): 2353 – 2360.

[9] Fu J, Sjodin P, Karlsson G. Loop-free updates of forwarding tables[J]. IEEE Transactions on Network and Service Management, 2008, 5(1): 22 – 35.

[10] Zhou W, Dong (Kevin) Jin, Croft J, et al. Enforcing customizable consistency properties in software-defined networks[A]. Proceedings of 12th USENIX Symposium on Networked Systems Design and Implementation[C]. San-

ta Clara, CA, USA: USENIX, 2015. 73 – 85.

[11] Reitblatt M, Foster N, Rexford J, et al. Abstractions for network update[J]. ACM SIGCOMM Computer Communication Review, 2012, 42(4): 323 – 334.

[12] Katta N P, Rexford J, Walker D. Incremental consistent updates[A]. Proceedings of ACM Special Interest Group on Data Communication Workshop on Hot Topics in Software Defined Networking [C]. Hong Kong, China: ACM, 2013. 49 – 54.

[13] Mcgeer R. A safe, efficient update protocol for OpenFlow networks [A]. Proceedings of ACM Special Interest Group on Data Communication Workshop on Hot Topics in Software Defined Networking[C]. Helsinki, Finland: ACM, 2012. 61 – 66.

[14] 刘艺,张红旗,杨英杰. 基于启发式调度的 OpenFlow 网络规则一致更新方案[J]. 电子学报,2017,45(7):1637 – 1645.
Liu Yi, Zhang Hong-qi, Yang Ying-jie. Consistent rule update scheme based on heuristic scheduling for OpenFlow networks[J]. Acta Electronica Sinica, 2017, 45(7): 1637 – 1645. (in Chinese)

[15] Mahajan R, Wattenhofer R. On consistent updates in software defined networks[A]. Proceedings of the Twelfth ACM Workshop on Hot Topics in Networks[C]. College Park, MD, USA: ACM, 2013. 20 – 26.

[16] McGeer R, Yalagandula P. Minimizing rulesets for TCAM implementation[A]. Proceedings of the 28th IEEE International Conference on Computer Communications[C]. Rio de Janeiro, Brazil: IEEE, 2009. 1314 – 1322.

[17] Galluccio L, Milardo S, Morabito G, et al. SDN-WISE: Design, prototyping and experimentation of a stateful SDN solution for WIreless SEnsor networks[A]. Proceedings of the 34th IEEE International Conference on Computer Communications[C]. Hong Kong, China: IEEE, 2015. 513 – 521.

作者简介

黄美根　男,1990 年出生,湖南娄底人,现为信息工程大学博士研究生. 主要研究方向为无线传感器网络、软件定义网络和网络安全.
E-mail: huang_meigen@163.com

郁　滨　男,1964 年出生,河南郑州人,现为信息工程大学教授、博士生导师,主要研究方向为信息安全、无线网络安全和视觉密码.
E-mail: byu2009@163.com