一种基于统计信息的通信芯片协议层

操作系统通信管理方案

发明人及申请人：张文通 张盛 重庆高开清芯科技发展有限公司

摘要

本发明针对通信芯片协议层操作系统的通信管理过程提出了一种新的消息预取及推送策略，通过统计分析节点历史消息推送、指令执行、与其它节点交互操作的日志对未来一段时间内有较大概率会使用的数据或指令进行预先请求和读取。通常在一个物联网系统内各节点相互通信所产生的业务消息流量是动态变化的，时高时低，有忙有闲，可以错开通信业务流量高峰时段，有针对性地向网络中的节点预取或推送数据的策略。目标是让各个节点提前获取到接下来的计算任务中可能会用到的数据、指令，存储在本地设备上，等到使用时无需发出请求便可即时使用，有助于提高节点工作效率，降低任务处理时延。若接下来使用的数据未被预取的数据命中，则采用传统的基于用户请求的方式发出请求报文，从服务方节点获取数据。若节点能在目标节点相对空闲、网络带宽利用率不高时进行预读取操作，将一部分数据和指令提前获取，当后续任务执行过程中产生真正的数据或指令需求时既可直接使用。这样既可以让节点本身的任务立即得到响应，使得处理过程更加实时高效，又可以提高CPU和系统带宽资源的利用效率，降低网络因通信业务量过大而出现拥塞的概率。

技术领域

本发明属于物联网领域，同时涉及数据挖掘、计算机体系结构、无线自组织网络、分布式系统、无线通信协议、边缘计算等技术领域。

技术背景

在通信芯片协议层操作系统内部，常见的消息推送模式为基于用户请求式的推送。基于用户请求的消息推送策略属于传统信息交互方式，当节点产生数据需求后，向拥有该数据的目标节点发出请求报文，目标节点收到报文后对该请求进行应答，发出请求的节点收到应答报文后进行解析，提取有效数据，这样即完成了一次数据请求-推送过程。该策略思路直观，实现简单，适合计算复杂度较低的嵌入式设备，但未考虑带宽利用情况、网络节点工作状态，若所请求数据非当前任务所需，实时性要求不高，且目标节点有大量任务待处理时，会给目标节点增加更多负担，消息推送过程缺乏灵活性。若有多个用户请求某一个用户或者某一种类型的消息报文，则出现网络拥塞导致推送效率降低的概率较大。

本发明提出的基于统计信息的消息预取及推送策略，通过节点交互日志挖掘节点之间在数据需求及数据获取方面的相似程度，是否存在关联。基于节点之间的关联性，提前发出数据请求或推送数据至目标节点。若猜测被命中则节点需要数据时既可直接使用，不再需要发出数据请求，从而降低任务处理耗时，同时减小网络通信业务高峰期发生网络拥塞的概率。

发明内容

本发明面向非同源节点交互过程中的数据获取方式，提出基于统计信息的消息预取及推送策略。非同源节点之间存在业务数据格式不统一或软硬件接口不一致的问题，可使用辅助通信模组实现交互。在通信模组的通信芯片协议层操作系统内部，定义统一业务数据格式，将节点设备状态、网络状态及功能指令信息封装为名片文件，通过交换名片文件实现节点间的交互操作及数据共享。本发明中的消息交互日志均基于名片文件实现。对于消息消息交互次数的统计也通过更新名片文件的对应字段实现。

对于网络中的某个具体节点而言，基于数据生产者的角度，节点通过自身所配置的传感器感知环境数据，是数据的提供者，可以基于统计信息将数据推送至其它节点；基于数据消费者的角度，节点执行动作或者做出决策任务还依赖从其他节点获取到的数据，可以基于统计信息进行提前预取。本发明中设计了根据消息预取及推送的历史记录有针对性地向网络中的节点预取或推送数据的策略。目标是让各个节点提前获取到接下来的计算任务中可能会用到的数据、指令，存储在本地设备上，等到使用时无需发出请求便可即时使用，有助于提高节点工作效率，降低任务处理时延。若接下来使用的数据未被预取的数据命中，则采用传统的基于用户请求的方式发出请求报文，从服务方节点获取数据。

物联网节点行为数据在系统中最简单的存在形式就是日志。系统在运行过程中都产生大量原始日志，每一条记录表示一次用户行为和对应的服务。比如光照强度、温度、湿度、节点工作状态、节点移动速度等，每一次服务请求和应答都会生成一个展示日志，其中记录了查询和返回结果。不同的节点对不同的业务信息请求频率和响应频率是不同的，必然有高频率的请求响应和低频次的请求响应，即不同节点间的联系紧密程度是不同的，如果各节点能够对自身的请求和被请求记录进行阶段性记录和分析，就能够发现不同节点之间的行为联系。据此可建立网络节点关联度列表，通过分析过去一段时间内网络各节点的历史消息推送记录，计算节点之间的关联度列表。当节点向网络中某服务方节点发出数据请求后，通过分析网络服务方节点关联度，也会向网络中与该服务方节点关联度较高的其它服务方节点发出数据请求。同理，当节点收到网络中某需求方节点发来的数据请求后，通过分析网络需求方节点关联度，向从而选择相似度高的消息请求，预取出来，实现消息的预取推送。

实现消息预取和推送的基础是建立网络节点需求关联度列表和网络节点服务关联度列表。这两个关联度列表以时间长度*T*为周期进行更新，列表中数值的计算任务由网络中的节点按照节点序号轮流执行。第*n*个周期的节点关联度列表由*m*号节点计算完成，则*n*+1个周期的节点关联度列表由*m*+1号节点计算完成。在第*n*个周期开始时，网络各节点将与自身进行数据预取和推送的节点标号及交互次数以报文的形式发送给*m*号节点。*m*号节点计算网络需求方节点关联度列表和网络服务方节点关联度列表，计算完成后将列表中与各节点相关的一部分计算结果分发给的对应的节点（每个节点会收到特定的一行和一列）。

若网络中有*N*个节点，则可统计网络节点间消息推送次数统计表如附表1所示：

表中推送节点指的是拥有数据的服务方节点，即数据的生产者；预取节点指的是有潜在数据请求的需求方节点，即数据的消费者。表中各统计值均以特定周期为单位，*Cxy*指的是某一个特定周期内节点*x*向节点*y*推送数据的次数。以节点1和节点2为例，计算二者同为需求方节点是的关联度，需要考虑网络其它节点(Node 3 - *N*)对节点1的数据推送记录与对节点2 的数据推送记录的相似程度，即*C*31, *C*41,…,*CN*1与*C*32, *C*42,…,*CN*2两个序列的相似程度。利用余弦相似度公式求出*D*12为



一般地，对可计算节点*x*和节点*y*需求**关联度为**



同理，计算节点*x*和节点*y*的服务关联度时，应考虑二者向网络其它节点推送数据记录的相似程度，可计算节点*x*和节点*y*服务**关联度为**

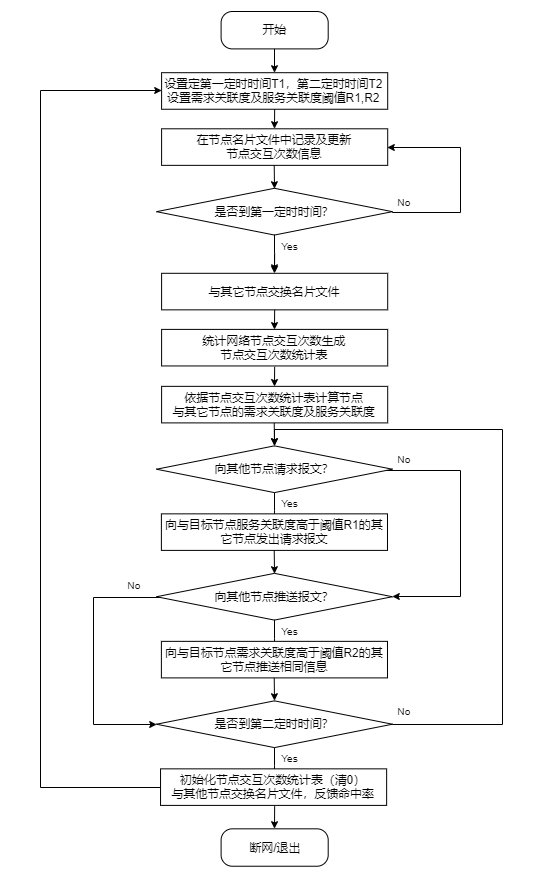


由上述**需求关联度和服务关联度**的计算公式计算得到网络需求方节点关联度列表和网络服务方节点关联度列表如附表2所示，以对角线为分割线，右上侧的三角部分为各节点之间的需求关联度，左下侧的三角部分为各节点的服务关联度。该表中数值的有效期为周期T，每隔T时间更新一次，表格数值由下一个节点重新计算。

实施方案

1. 设置两个周期值T1和T2，T1表示消息预取和推送策略应用前，统计消息交互记录的时长，T2表示消息预取和推送策略的应用时长。T1+T2为整个算法运行的周期；
2. 设置节点需求关联度阈值R1及服务关联度阈值R2；
3. 0-T1时间段内，记录并更新节点与其它节点之间的交互次数，不包含有效数据的请求报文及应答报文不计入在内（判别标准是报文长度大于数据报文头部的长度，包含有效数据载荷）；
4. T1时刻，节点将与其他节点的交互次数写入名片文件，并与其他节点交换包含交互信息的名片文件，根据交互信息生成节点交互次数统计表。依据交互次数计算与其它节点的需求关联度与服务关联度；序号为 *t*的节点计算结果为附表2的第*t*行以及第*t*列；
5. T1-T2时间段内，节点A向节点B推送数据时，节点A会向与节点B需求关联度高于阈值R1的其它节点推送相同数据；节点A向节点C请求数据时，节点A会向与节点C服务关联度高于阈值R2的其它节点发出相同的数据请求。节点名片文件记录收到的预先向其他节点获取或预先被推送至本节点的报文数目。若提前推送或预取的报文恰好是本节点需要的，则猜测被命中。
6. T2时刻，节点将网络状态信息表清0，根据T1-T2时间段内名片文件记录的报文命中情况及反馈的命中率信息，更新反馈需要调整定时器及阈值。如果命中率较高但出现网络拥塞情况，则提高R1与R2，减少发起预推送或预请求的次数。如果命中率较低且网络无明显拥塞发生，则可增大T1，降低R1与R2，进入下一个周期。

附图及附表说明



附图1 消息预取及推送策略流程图

附表1 网络节点间消息推送次数统计表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 预取节点  推送节点 | Node 1 | Node 2 | Node 3 | … | Node *N* |
| Node 1 |  | *C*12 | *C*13 | … | *C*1*N* |
| Node 2 | *C*21 |  | *C*23 | … | *C*2*N* |
| Node 3 | *C*31 | *C*32 |  |  | *C*3*N* |
| … | … | … |  |  | … |
| Node *N* | *CN*1 | *CN*2 | *CN*3 | … |  |

附表2 网络需求方节点关联度列表和网络服务方节点关联度列表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 节点2  节点1 | Node 1 | Node 2 | Node 3 | … | Node *N* |
| Node 1 |  | *D*12 | *D*13 | … | *D*1*N* |
| Node 2 | *S*21 |  | *D*23 | … | *D*2*N* |
| Node 3 | *S*31 | *S*32 |  |  | *D*3*N* |
| … | … | … |  |  | … |
| Node *N* | *SN*1 | *SN*2 | *SN*3 | … |  |