July 2023

DETERMINISTIC INFORMATION PULL AND AGGREGATION IN A DESTINATION-ORIENTED DIRECTED ACYCLIC GRAPH (DODAG) TOPOLOGY

Pascal Thubert

Jerome Henry

Patrick Wetterwald

DETERMINISTIC INFORMATION PULL AND AGGREGATION IN A DESTINATION-ORIENTED DIRECTED ACYCLIC GRAPH (DODAG) TOPOLOGY

AUTHORS:
Pascal Thubert
Jerome Henry
Patrick Wetterwald

ABSTRACT

Techniques are presented herein that support a deterministic collection method in the form of a scheduled pull mechanism, whereby a collection point may compute a schedule and then distribute the same so that all of the collecting nodes and network nodes can optimally transmit, aggregate, and relay information to the collection point. Under aspects of the presented techniques, a distributed set may be organized as a Destination-Oriented Directed Acyclic Graph (DODAG) topology to the collection point. The above-described schedule may then be generated whereby the DODAG parents emit only after all of their children (and, recursively, all of their descendants) have emitted. Under such an approach, an application-aware parent can aggregate (or factorize) the data and consume minimal bandwidth. A second schedule may be generated just for missing data, accounting only for the sources where information was lost, so that retry operations comprise smaller schedules.

DETAILED DESCRIPTION

In a situation that requires the sending of data that originates at roughly the same time from a plurality of distributed wireless nodes to one central collection point, a number of effects typically appear. Such effects may include a collision between transmissions, a clogging of the links that aggregate traffic, unbounded delays to a retry operation, large numbers of losses, etc.

Several real-world examples will help to frame the above-described context and illustrate the above-described effects. A first example encompasses alerts that are emitted by multiple nodes that observe the same event as in, for example, a Generic Object-Oriented Substation Events (GOOSE) network.

A second example encompasses decentralized autonomous organization (DAO) messages in a Routing Protocol for Low-Power and Lossy Networks (RPL) network. In the current specification for such an environment (i.e., Internet Engineering Task Force (IETF) Request for Comments (RFC) 6550) the storing mode DAO messages are sent asynchronously by all of the nodes, with each node providing a 'dot' of a 'picture' that is taken at a different time. Ideally, the root would take a full picture of the network at a desired instant. However, that would require a pulling of all of the DAO messages, something that the protocol does not explicitly support (even though such support was desired) due to the risk of a collision.

A third example encompasses a time measurement in a wide area or a large cluster. For example, in ultra-wideband (UWB) technology (as specified by the FiRa Consortium (FiRa) standards), a cluster of depth n is formed between m anchors and one anchor is elected as the primary anchor (PA). The PA sends a timestamped trigger message to which all of the other anchors should respond, with such a response message containing the arrival time of the PA message and the departure time of the response message. Such a process is used to synchronize the clocks for all of the anchors, as location tracking in such an environment relies on a time difference of arrival (TDoA) methodology. While some anchors may be in direct radio frequency (RF) range of the PA, others may not. In such a case, the anchors at depth p may receive the PA message as it is relayed by another anchor at depth p-1. A depth-p anchor may respond to an anchor at depth p-1 and that anchor may relay the depth-p anchor's timestamped response message upstream until it reaches the PA. The unsolved difficulty concerns organizing the messaging structure to avoid collisions while making sure that all of the messages from all of the anchors reach the PA within each measurement cycle.

In short, the goal of deterministic networking is to provide high reliability within a bounded latency. However, the above-described effects must be overcome to enable reliable deterministic data collection.

Techniques are presented herein that support a deterministic collection method in the form of a scheduled pull mechanism, whereby a collection point may compute a schedule and then distribute the same so that all of the collecting nodes and network nodes can optimally transmit, aggregate, and relay information to the collection point.

Under the presented techniques, the multi-source emissions in a pull model (such as the DAO message and time measurement examples that were described above) may be scheduled, with such a schedule accounting for possible aggregation on the way to a collection point. In part, when an event is detected, which typically is a push model from a source to a controller, a central controller is first informed of the same by one, or a few, of the sources. The controller may then compute a schedule to obtain the needed information from all of the possible sources. Among other things, the presented techniques may employ a limiting mechanism (such as, for example, the common Trickle algorithm (as described in IETF RFC 6206)) in the affected broadcast domains.

After a controller becomes aware that there is a data collection to be made, the controller may, according to the presented techniques, perform a series of steps.

A first step encompasses an enumeration of the router links in a breadth-first order. In brief, a controller becomes aware of the Destination-Oriented Directed Acyclic Graph (DODAG) topology and then walks that topology breadth first, meaning that it counts, by the order of the cost of a node, to the root through the hop (e.g., a hop count if all of the costs are one, using any order for an equal cost). If there is a sense of an east-west orientation, then such counting may begin on the eastern side in order to interleave space between a child and a parent. Figure 1, below, illustrates elements of the above-described process.

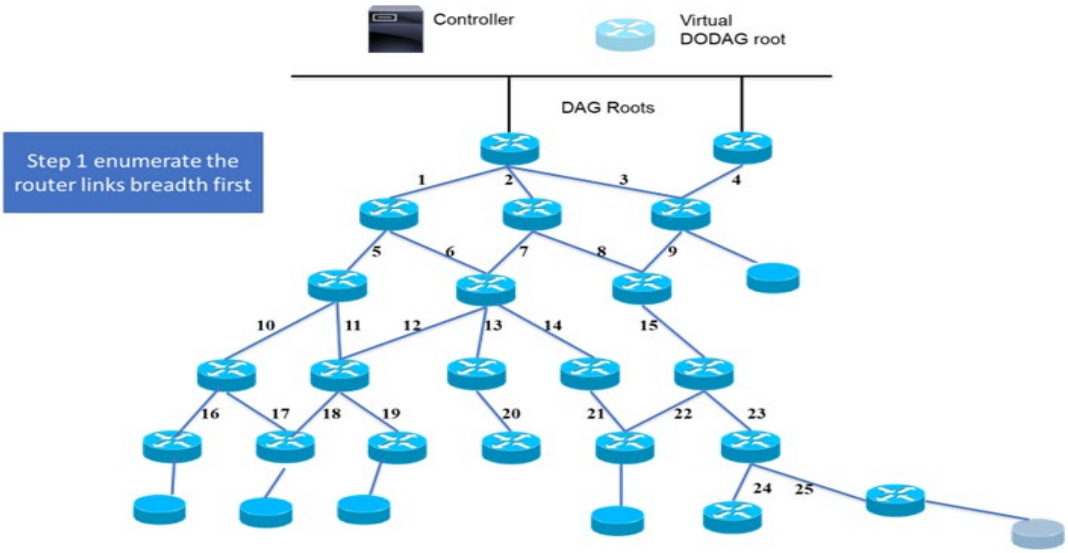


Figure 1: Breadth-First Router Link Enumeration

The result of the above-described process is a strict order (as shown in Figure 1, above) with the desired property that a child has a higher number than its parent. According to the presented techniques, it is possible to number an entire DODAG topology or just a preferred parent tree. In the former case, the same redundancy is obtained as path redundancy exists. The data may be duplicated until a common aggregation point that will filter the duplicates, effectively performing deterministic networking (DetNet) packet replication, elimination, and ordering functions (PREOF) but at the data or information level.

To realize a schedule order, during a second step an array containing the counting results from step one, above, may be reversed, yielding an order in which a child appears before their parent, as shown in Figure 2, below.

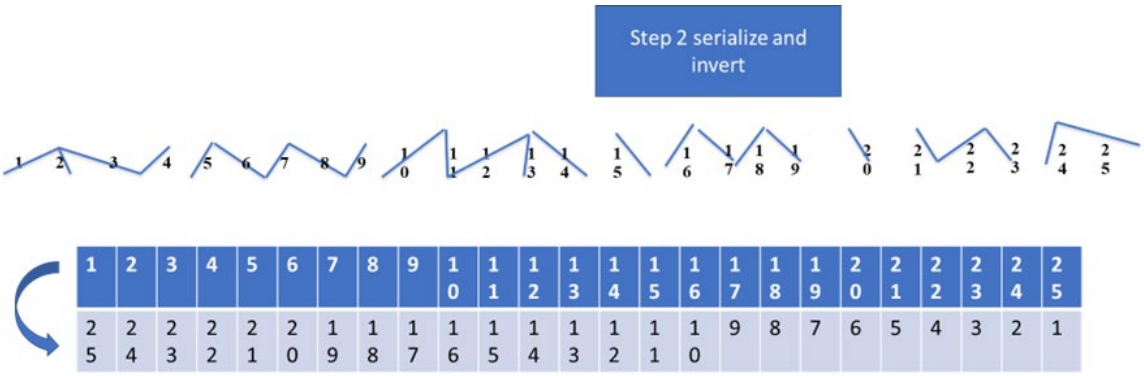
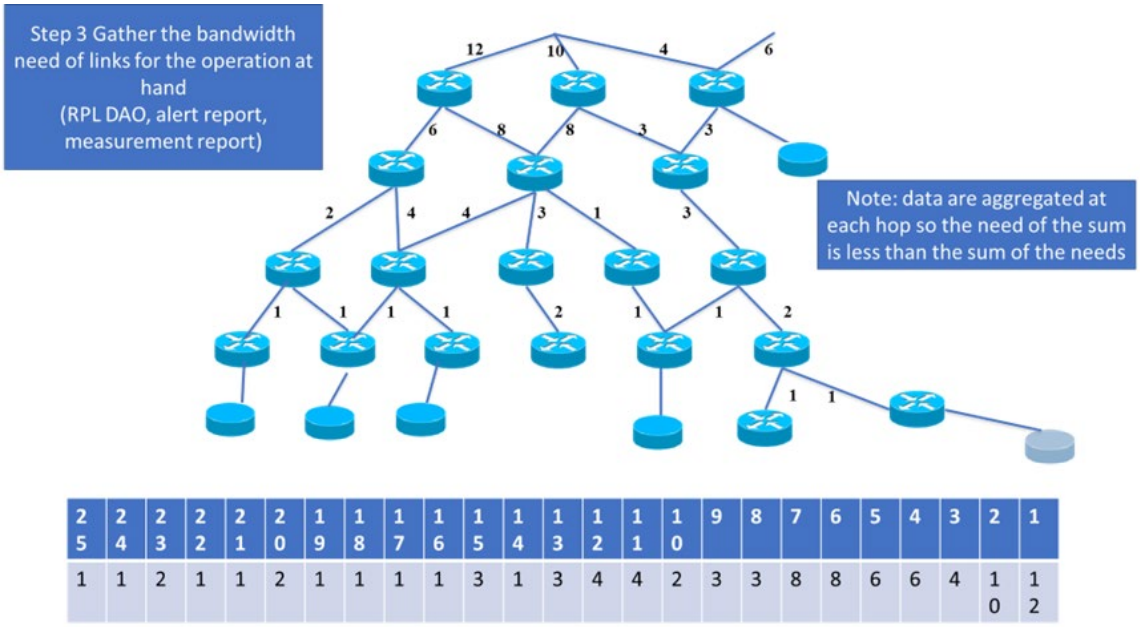


Figure 2: Serialization and Inversion

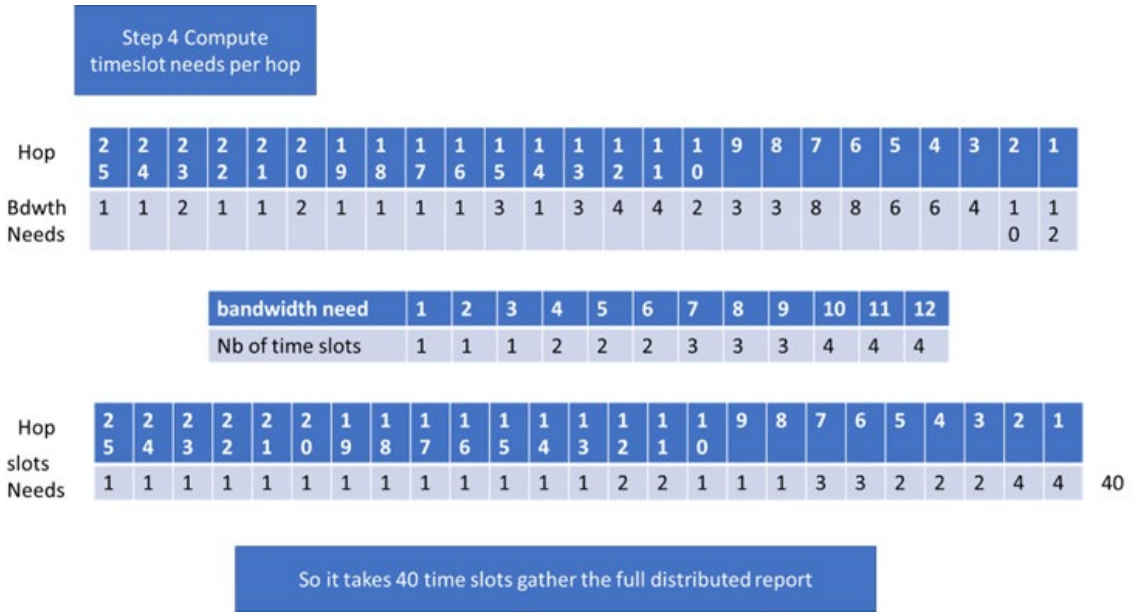
A third step encompasses gathering the bandwidth that is required by each link for the operation at hand (such as an RPL DAO message exchange, the sending of an alert report, the sending of a measurement report, etc.). This accounts for the potential aggregation of information (where several data elements come in but only one data element goes out) where multiple data records can be concatenated into one packet, or the same information is filtered and counted so that it may be reported only once but with a counter and possibly a list of sources. It is important to note that data are aggregated at each hop, so the need of the sum is less than the sum of the needs. Figure 3, below, depicts elements of the above-described process.



25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
1	1	2	1	1	2	1	1	1	1	3	1	3	4	4	2	3	3	8	8	6	6	4	10	12

Figure 3: Gathering Bandwidth Needs

A fourth step maps a need for bandwidth to a timeslot consumption table (accounting for the possibility that a timeslot is indivisible and may host more than one packet), as shown in Figure 4, below.



Hop	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
Bdwth Needs	1	1	2	1	1	2	1	1	1	1	3	1	3	4	4	2	3	3	8	8	6	6	4	10	12

bandwidth need	1	2	3	4	5	6	7	8	9	10	11	12
Nb of time slots	1	1	1	2	2	2	3	3	3	4	4	4

Hop	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
slots Needs	1	1	1	1	1	1	1	1	1	1	1	1	1	2	2	1	1	1	3	3	2	2	2	4	4

40

Figure 4: Per-hop Timeslot Need Computation

During a fifth step, the number of required timeslots may be counted (which, as depicted in Figure 5, below, is 40 for the instant example) and a schedule may be generated, with such a schedule inserting a blank if necessary to ensure that there is at least one timeslot between the last child transmission and the parent transmission to allow for an aggregation computation.

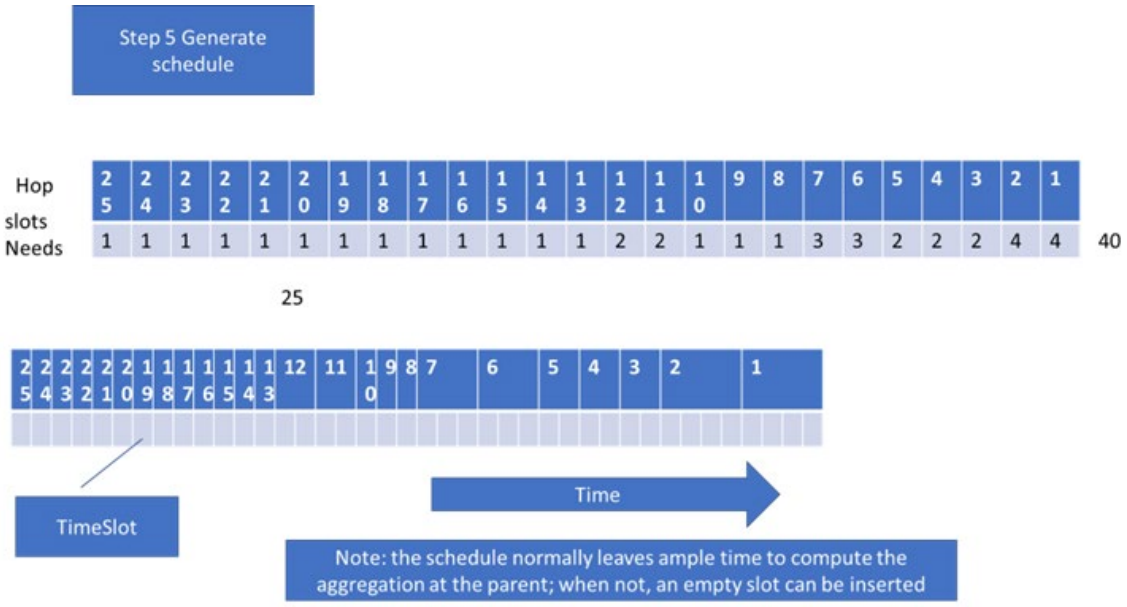


Figure 5: Schedule Generation

Under the presented techniques, all of the nodes are expected to respond, even with blank information when a node has nothing to report. Since an aggregator indicates the list of aggregated sources, the result is that a controller knows when a piece of information is missing.

A sixth step encompasses a retry operation to pull any missing information. Under this step, the process that was described in the above steps may be rerun, but accounting only for the sources that are missing. Figure 6, below, depicts elements of this process.

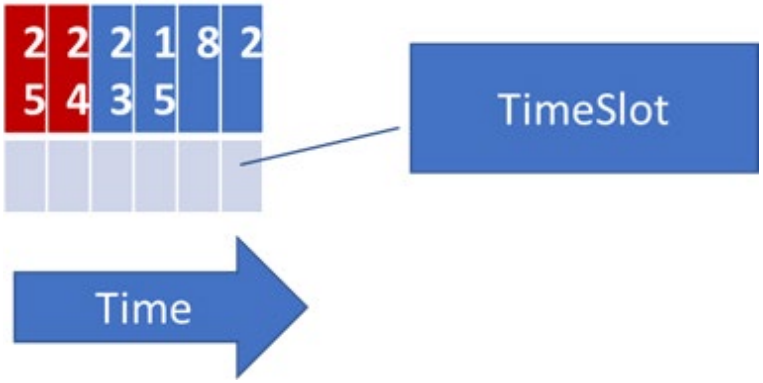


Figure 6: Retry Operations

As depicted in Figure 6, above, the transmissions over links 24 and 25 were missing (perhaps they experienced some type of local interference). Accordingly, a retry schedule may include only those sources and the path along the referred parent tree.

While retry operations (as described above under step six) resolve a short-term interference, they do not address structural collisions. A seventh step deals with structural collisions, where it is better to include some retry capabilities inside of the schedule itself. For example, a node may make use of timeslots that are between its transmission and its parent's transmission for retries, and only in the case of a failure in a retry schedule may step six be invoked. For this, the controller may provide to the node the best timeslots based on the least chance of interference between a transmission from the timeslot owner to its parent on one hand, and the instant node and its parent on the other hand, using a Bayesian estimation of the risks of collisions. This means that the additional chance for the packet to pass is also a chance for the other packet to be interfered with, with a risk of a cascading effect. If there are not enough timeslots to safely accommodate the retries (e.g., if the same timeslot must be provided as a backup for too many other nodes) then the controller may insert new timeslots into the schedule that are to be used only for retries and which are preferred over collisioning ones.

Under an eighth step, as identified timeslots continue to be missed between iterations a controller may perform a Bayesian estimation of the risks of collisions between timeslots and then redistribute the timeslots accordingly, as depicted in Figure 7, below.

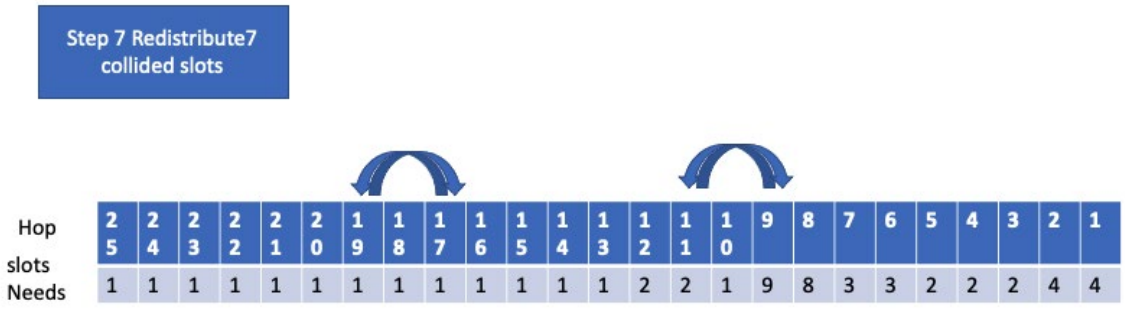


Figure 7: Redistribution of Timeslots

The above-described step may be repeated until all of the collisions are resolved. Further, in a large system there may be more nodes than timeslots. Accordingly, as a schedule (as described under step five, above) surfaces more timeslots than what are available, step seven, as described above, may be instantiated to allow distant groups and nodes to communicate during the same timeslot.

As described and illustrated above, the techniques presented herein support the computation and distribution of an optimized schedule for a deterministic data collection. However, it is important to note that those techniques may be reversed to support a deterministic periodic push of recurring, but node-dependent, information to a set of nodes.

The presented techniques may be deployed in a number of different contexts. One such context encompasses a mesh environment, such as a Wireless Smart Utility Network (Wi-SUN) environment, where the information that is collected is only valid for a limited amount of time and thus must be delivered before the expiration of that time. One example of a Wi-SUN environment, which applies to a smart grid setting, is power control. Such an environment collects instantaneous power consumption information for a collection of households and decides which households can start additional appliances within a power budget. The same logic could apply to a mesh environment that implements Time-Sensitive Networking (TSN) at each hop, even though such an approach has not yet been built. On the other hand, the International Society of Automation's (ISA's) ISA100.11a standard, the wireless highway addressable remote transducer protocol (WirelessHART), and Wi-SUN all exist, are all deployed, and may all leverage the techniques presented herein.

In summary, techniques have been presented herein that support a deterministic collection method in the form of a scheduled pull mechanism, whereby a collection point may compute a schedule and then distribute the same so that all of the collecting nodes and network nodes can optimally transmit, aggregate, and relay information to the collection point. Under aspects of the presented techniques, a distributed set may be organized as a DODAG topology to the collection point. The above-described schedule may then be generated whereby the DODAG parents emit only after all of their children (and, recursively, all of their descendants) have emitted. Under such an approach, an application-aware parent can aggregate (or factorize) the data and consume minimal bandwidth. A second schedule may be generated just for missing data, accounting only for the sources where information was lost, so that retry operations comprise smaller schedules.