Data Aggregation Design:

1. Data aggregation will be performed at the source only.
2. A Data Message sent in a particular gradient contains data matching at list one Interest of that gradient.
3. Intermediate nodes forward the packet. They do not split or merge Data Messages.
4. Intermediate nodes send one Data Message per matching gradients.
5. The Matching Rules are such that Data Message matches an Interest if at least a portion of it, matches in a one-way-match.
6. The sink is responsible for filtering the its relevant data, as a Data Message might contain additional mismatching data.

Reasoning:

1. Splitting Data Messages in intermediate nodes results in duplicate Data if the split Messages happen to meet in the same node. See figure 1.
2. If the packet them self are marked by a unique identifier to suppress duplicate data might be lost. See figure 2.
3. The only way to prevent duplicates and data loss at the same time is to uniquely identify every data element. This is not a reasonable solution since the cost in data size and processing is higher than the aggregation gain.
4. In order to aggregate Data Messages at intermediate nodes verses source nodes, data must be cached until another matching Data Message arrives. This is a reasonable solution which introduces memory space and computation cost. A basic requirement for this aggregation is that one of two matching Data Messages has enough space to also store the attributes of the other. Since data aggregation is already done optimally in the source in coordination with the source Data Sample Manager, it is unlikely to happen. Note: data is identified by a few attributes; each attribute is 4 bytes long.

Data Message Packet Size:	60 bytes. (Intuition – Currently its 30 bytes)
Data Message Header Overhead:	10 bytes
Data Space Left:	50 bytes
MAX Number of Attributes:	12 attributes

Note: Data Type, 2D location, Time Stamp and Interval requires 5attributes. Sample Manager of 7 data samples will already occupy the complete packet size.

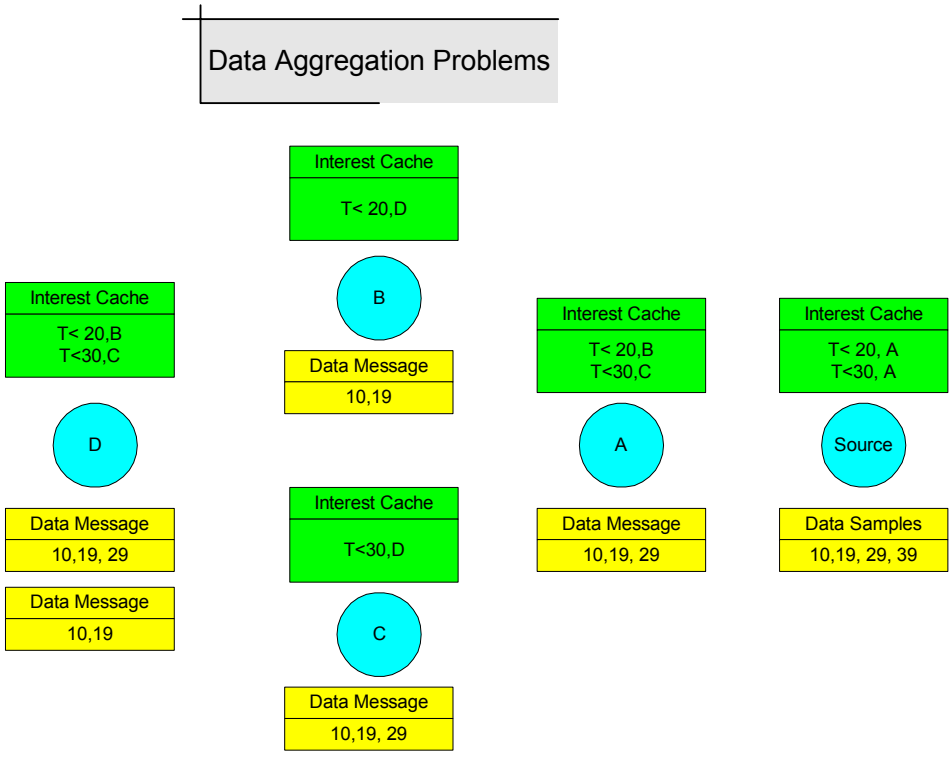

Figure 1: Data Splitting: Data is pulled up stream from source to D. Note data splitting at A results in duplicate data at D.

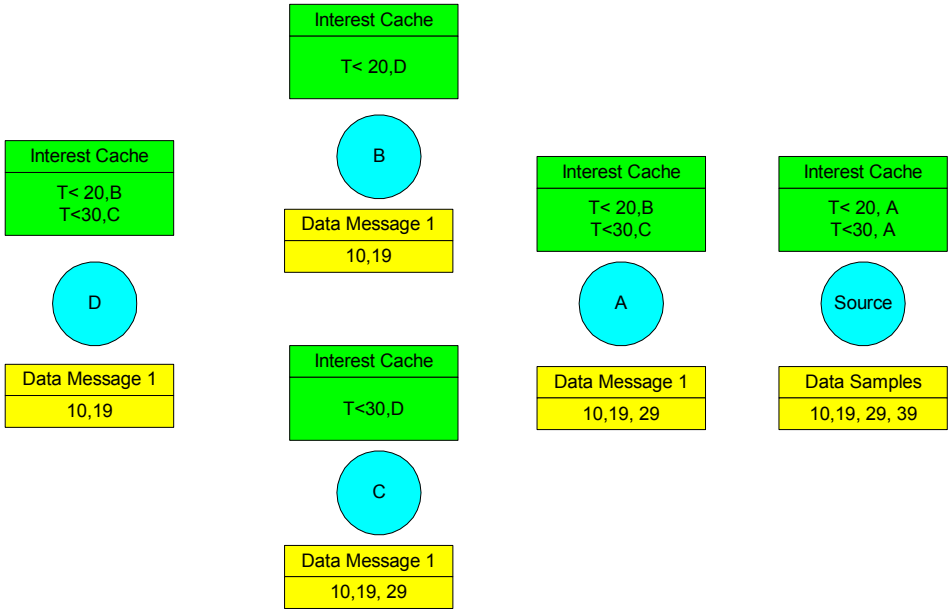

Figure 2: Data Splitting with ID. Data is pulled up stream from source to D. Note data splitting at A results in data loss (29) at D assuming C's packet was droped since duplicate.