Adaptive Data Replication in Real-Time Reliable Edge Computing for Internet of Things

Chao Wang, Christopher Gill, Chenyang Lu

What is being solved?

- For fault tolerance we need to replicate data.
- Replicating at the speed of arrival is inefficient.
- To come up with an adaptive data replication architecture for IoT edge computing that can meet applications’ latency and data-loss requirements with efficiency.

Challenges

- Sensing devices have limited storage capacity
- Limited network bandwidth of IoT gateways. Need to consider while deciding data replication.
- Applications have restrictions such as
  - Can tolerate only a certain number of data loses
  - End to end timing requirements

System Model and Analysis

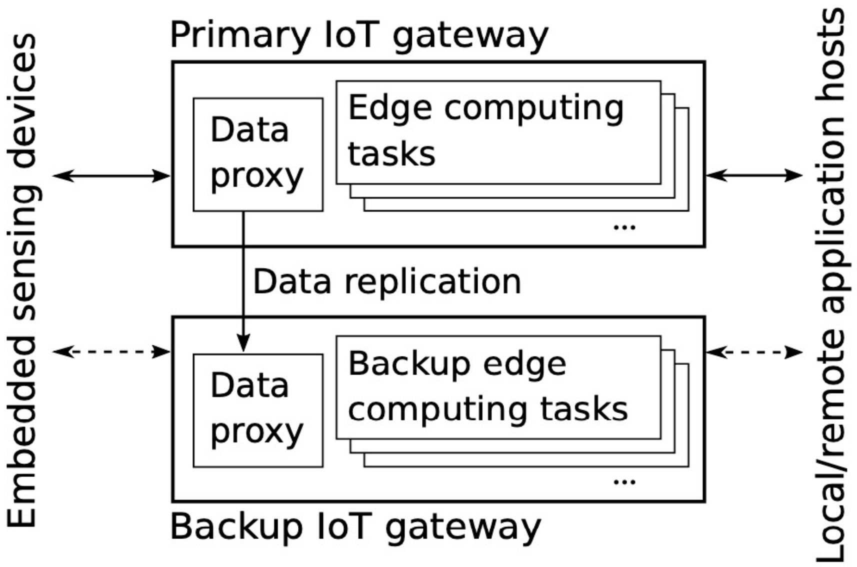


Fig. 1. Edge computing for Internet of Things.

Publish subscribe model

System Model and Analysis

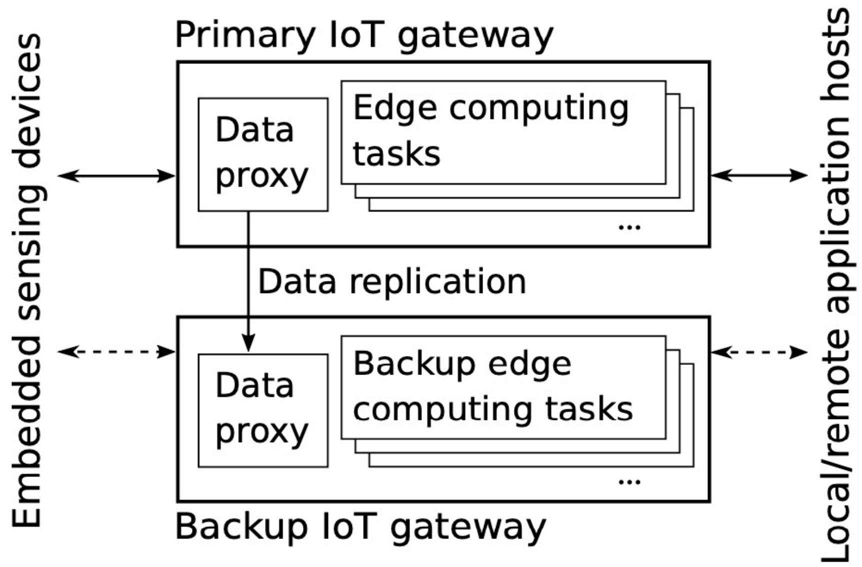


Fig. 1. Edge computing for Internet of Things.

Publish subscribe model

Key terms and Notations

- Data topics
- T_i - Minimum inter-publishing time for data topic I
- N_i – Data elements that a publisher can keep for a data topic I
- L_i – Maximum number of consecutive losses that a subscriber can accept for a data topic i.
- D_i^p – latency requirement – soft end-to-end deadline.
- D_i^r – relative replication deadline

System Model and Analysis

TABLE I
EXAMPLE DATA TOPIC SPECIFICATION.

Category	L_i	N_i	D_i^p (ms)	T_i (ms)
1	0	1	∞	50
2	0	1	100	100
3	0	1	500	500
4	3	0	50	50
5	3	0	100	100

Observations that were used to come up with the categories:

- Data publishers have limited data storage for re transmission
- Data topics may have moderate or no loss-tolerance requirements – inference tasks where data loss can be compensated by estimation.
- Data topics may require zero loss but have no latency requirement - logging

When do we actually need to replicate data?

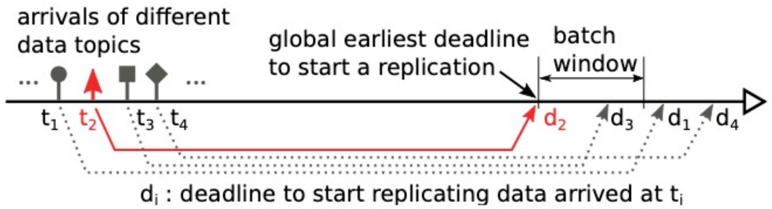
- Let $x_i(t)$ be the largest number of consecutive uncovered data elements, the system meets fault tolerance if $x_i(t) <= L_i$ at all times.
- For small N_i : $x_i(t)$ depends on :
  - T_i
  - Edge computing task time
  - Scheduling
- No replication needed if elements are processed before new elements are sent if $L_i >= 1$.
- Regular data replication can be used based on some predetermined conditions.

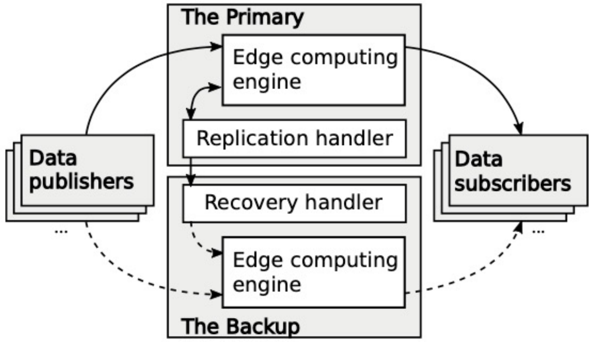
When to do the replication?

- A deadline is decided
- 2 lemmas help us understand the constraints better and come up with an architecture:
  - *For data topic* i*, to prevent more than* L_i *consecutive data losses,* L_i *and* N_i *cannot be both zero.*
  - *For data topic* i*, set parameter* $M_i \geq 1$ *and let* $y = L_i - M_i$*. To prevent more than* L_i *consecutive data losses, the replication deadline must satisfy the following bound:*
    - $D_i \leq (N_i + y + 1)T_i - T_{FO} - \delta_{PP} - \delta_{PrB}$
      - T_{FO} : Fail over time
      - δ_{PP} : Latency from publisher to Primary
      - δ_{PrB} : Latency from Primary to Backup
    - Lemma 2 implies that a shorter interval between replications (a smaller M_i) can permit a longer replication deadline.

ARREC Architecture

Mark → Wait → Clear → Batch





Empirical Analysis

- Two M_i configurations for AAREC – 1 and L_i : Two extremes
- Baseline:
  - Retransmission-only : No replication. Used to understand overhead of replication.
  - Periodic : 50ms (shortest in topic categories) and 25ms

Empirical Analysis