SpanEdge: Towards Unifying Stream Processing over Central and Near-the-Edge Data Centers

Hooman Peiro Sajjad[*], Ken Danniswara[*], Ahmad Al-Shishtawy[†], Vladimir Vlassov[*]

* KTH Royal Institute of Technology

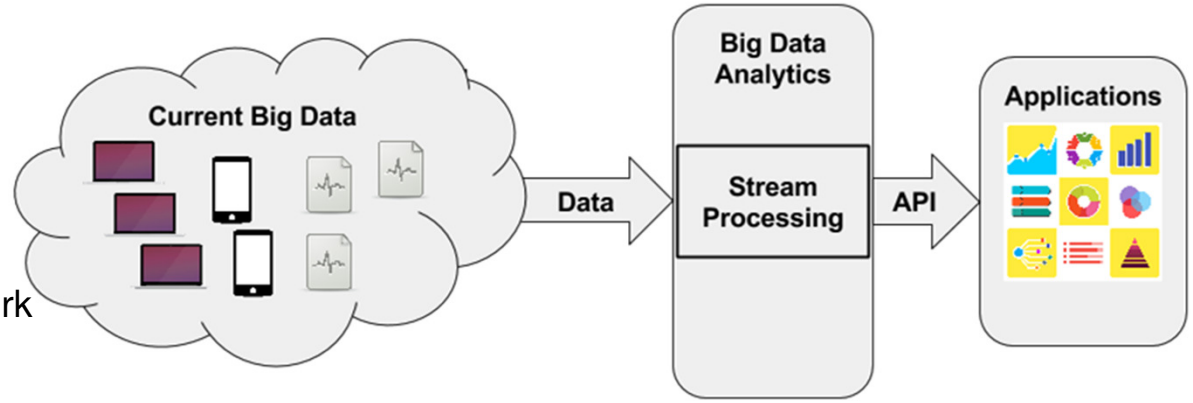
† SICS Swedish ICT

SEC 2016
Washington DC, USA

Real-time Analytics

Examples:

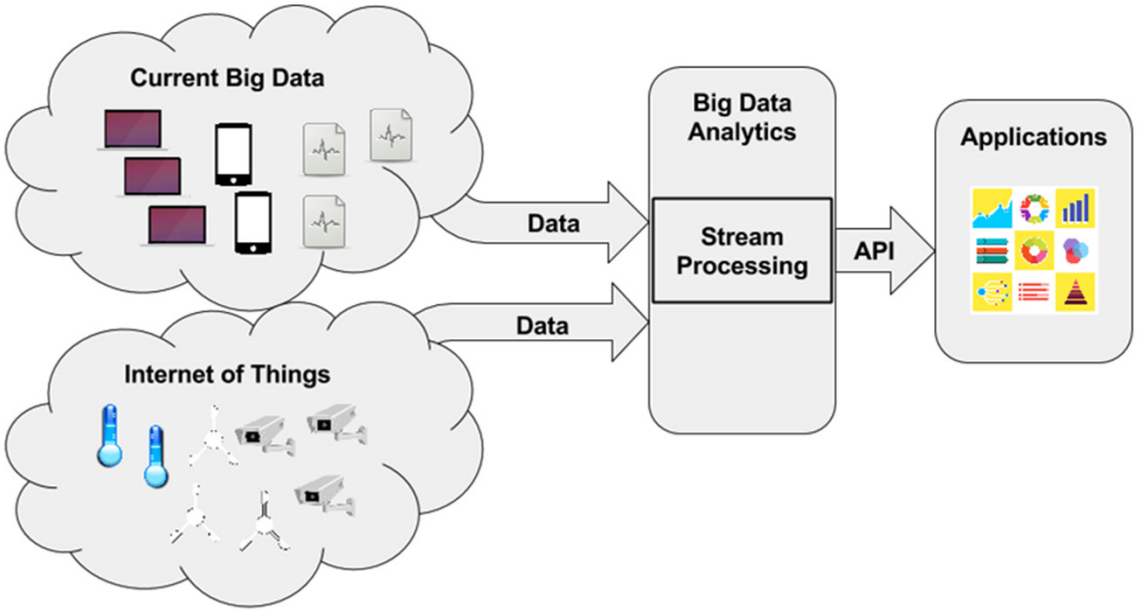
- Server logs
- User clicks
- Social network interactions



Real-time Analytics

Examples:

- Server logs
- User clicks
- Social network interactions

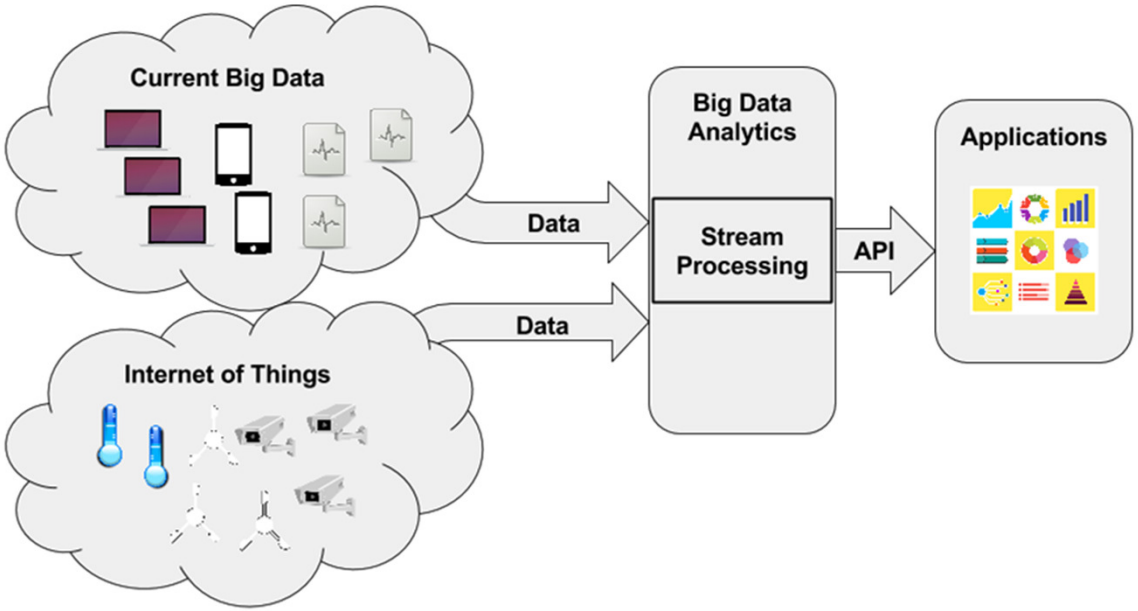


Real-time Analytics

Examples:

- Server logs
- User clicks
- Social network interactions

50 billion devices connected to the Internet by 2020



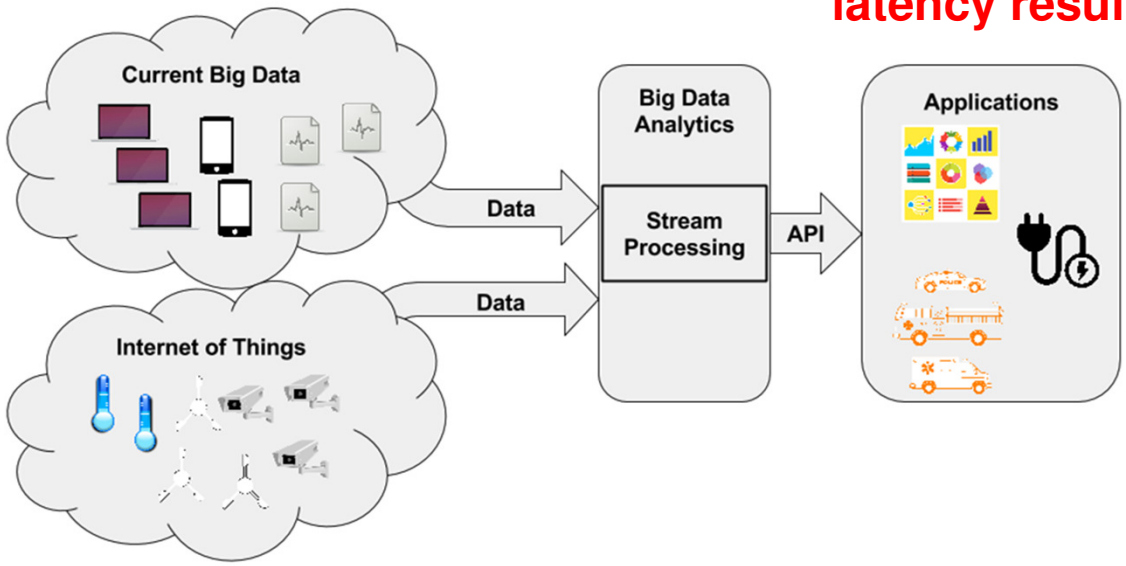
Real-time Analytics

New consumers of data analytics are joining the Cloud that require low-latency results

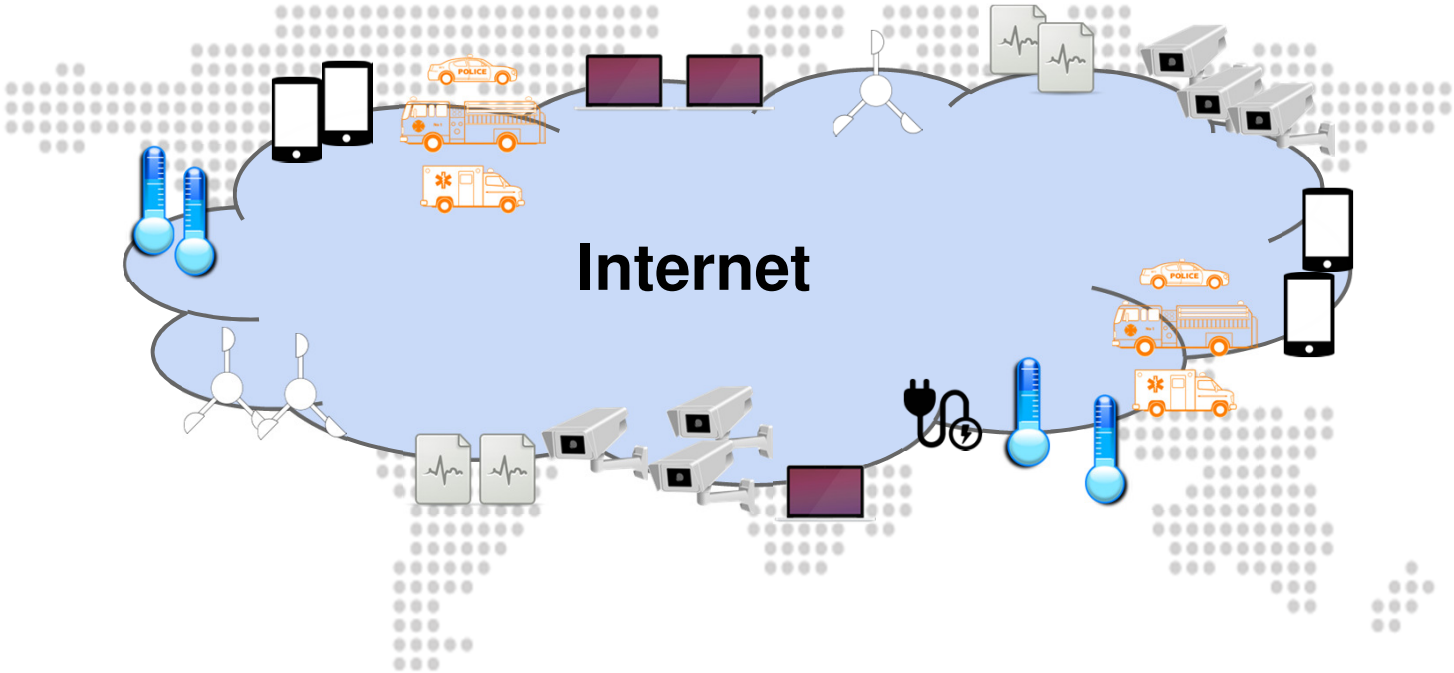
Examples:

- Server logs
- User clicks
- Social network interactions

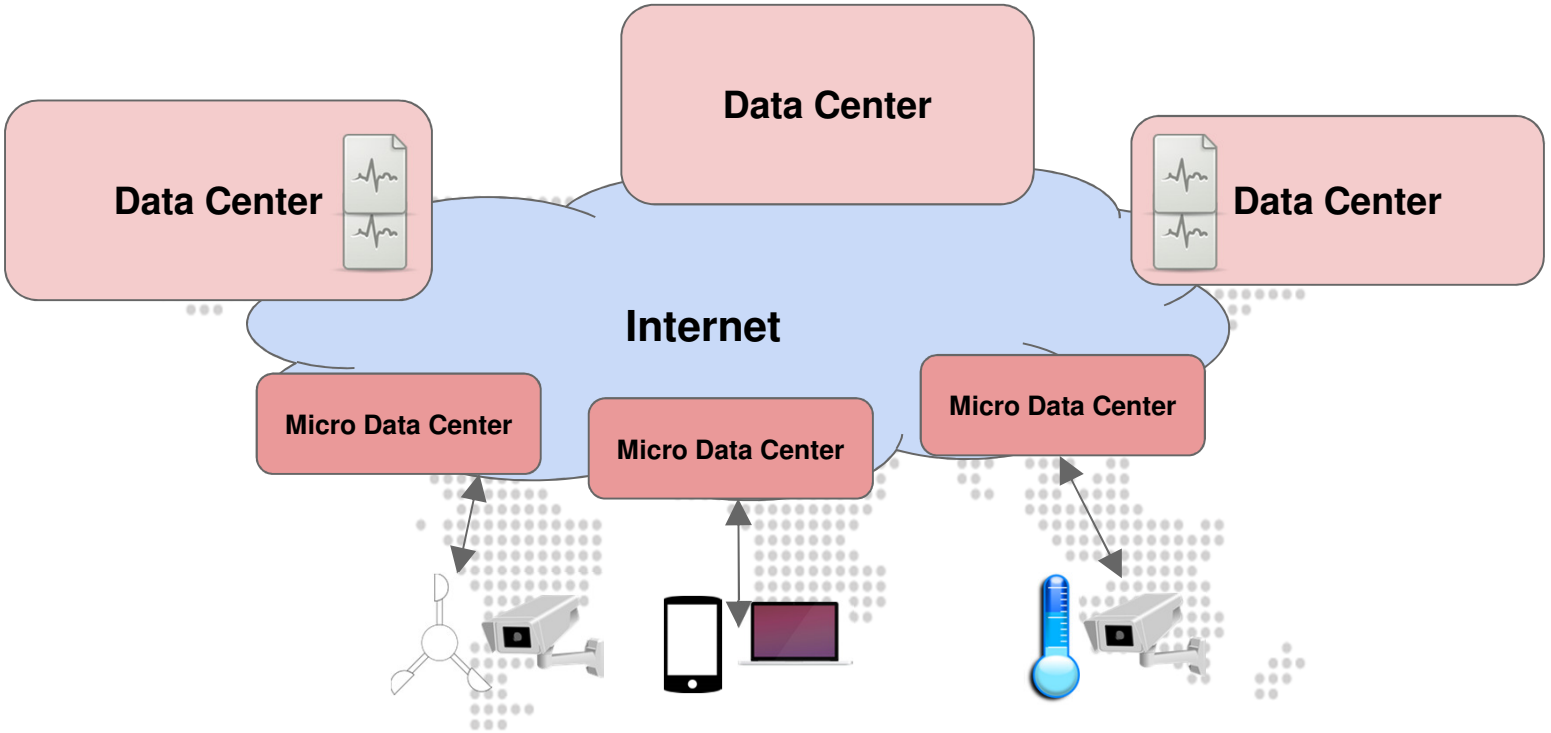
50 billion devices connected to the Internet by 2020



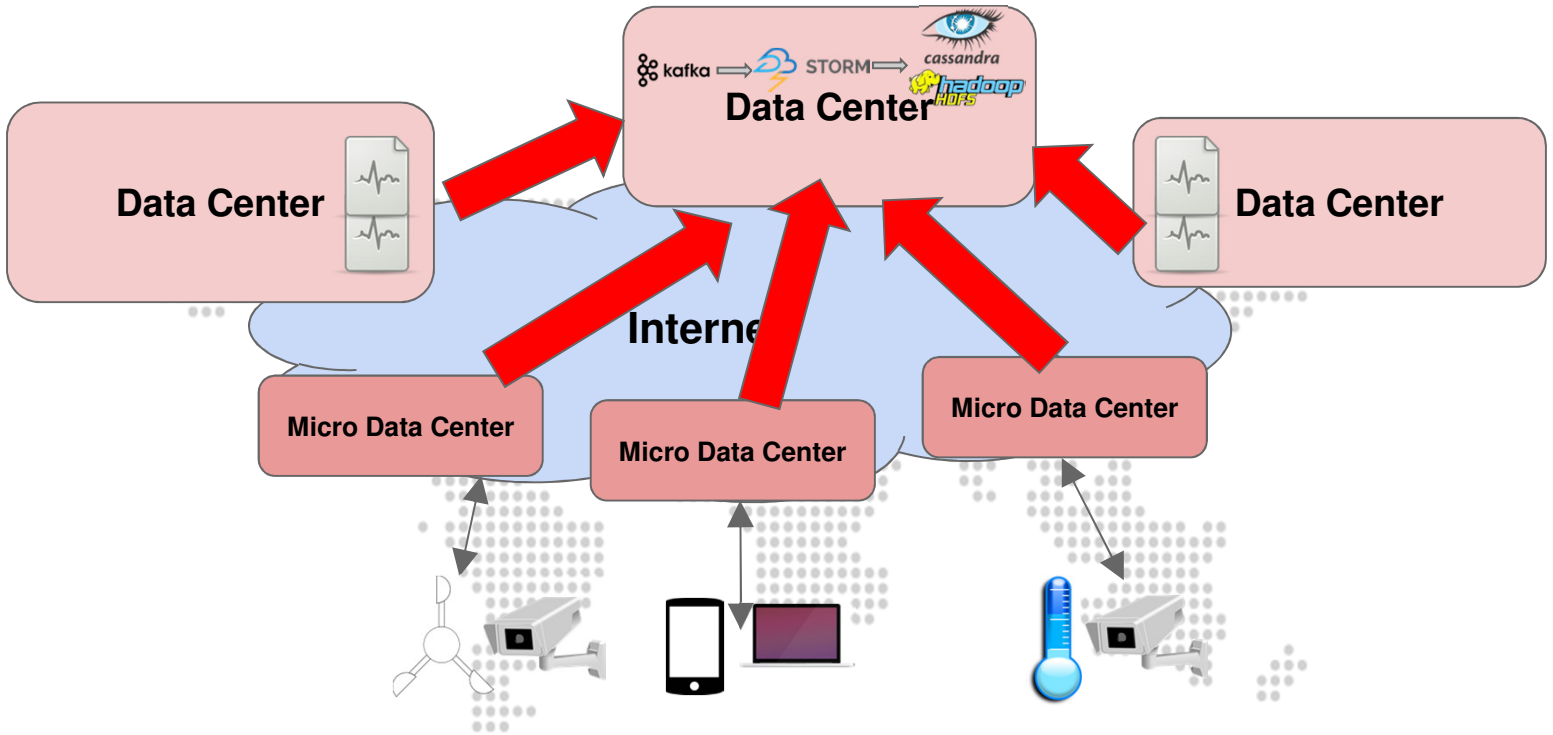
Geo-Distributed Data



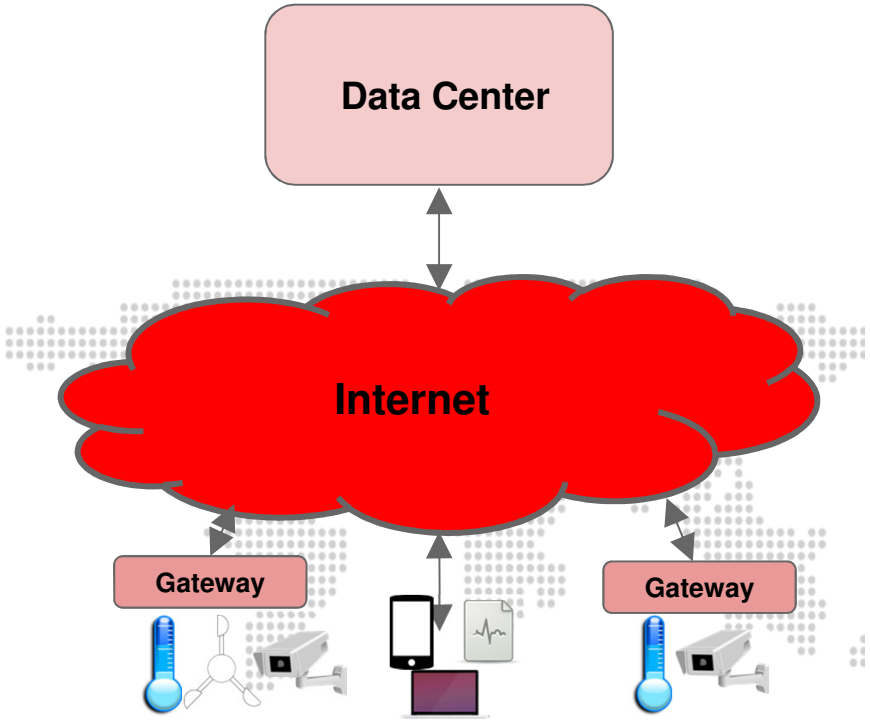
Central Approach



Central Approach

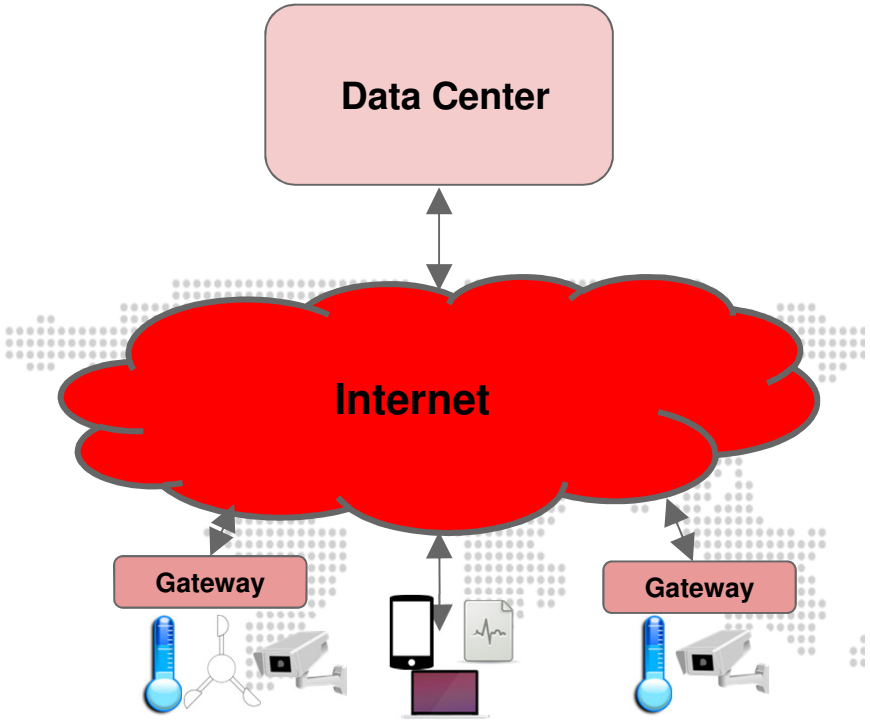


Problems: Wide Area Network



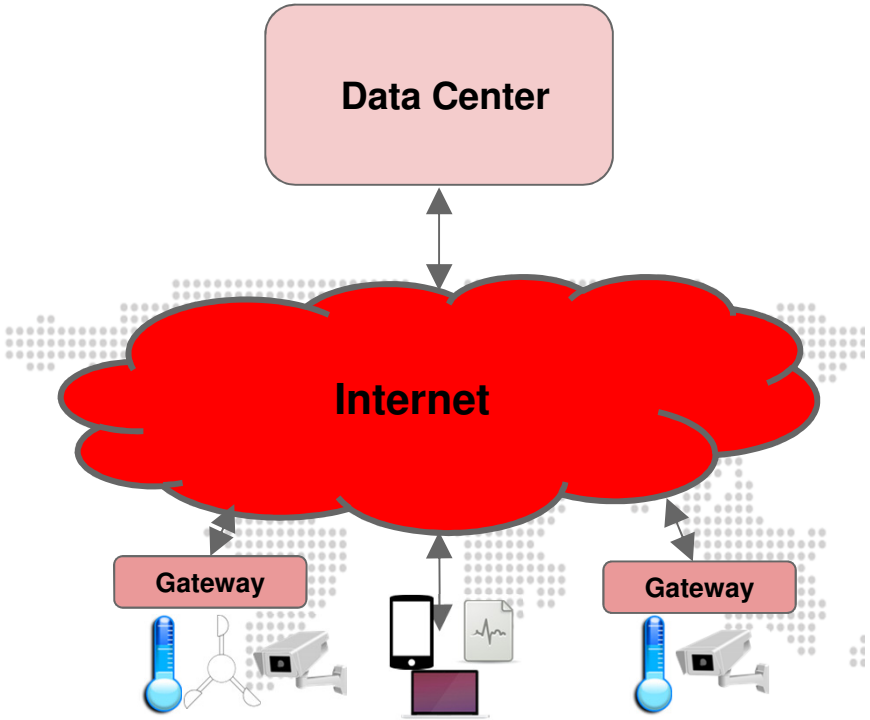
Problems: Wide Area Network

- The WAN bandwidth is scarce and expensive.



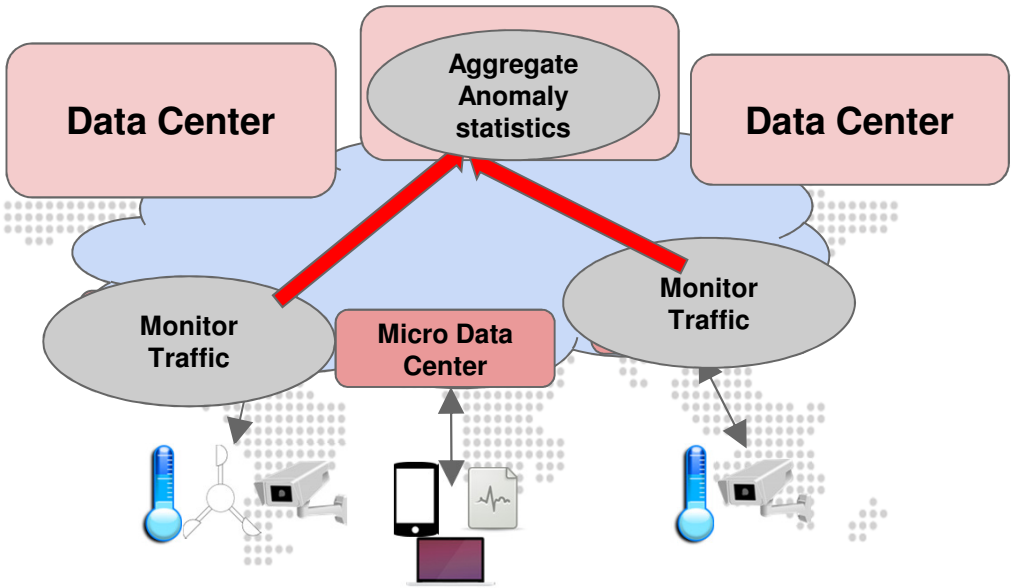
Problems: Wide Area Network

- The WAN bandwidth is scarce and expensive
- Long communication latency over the WAN links



Problems: Hard to Program

- It is hard to program and maintain stream processing applications both for the edge and for central data centers



Problem Definition

How to enable and achieve effective and efficient stream processing given the following:

- Multiple central and near-the-edge DCs
- Multiple data sources and sinks
- Multiple stream processing applications

and:

- Data is streamed from sources to their closest near-the-edge DC
- DCs are connected with heterogeneous network

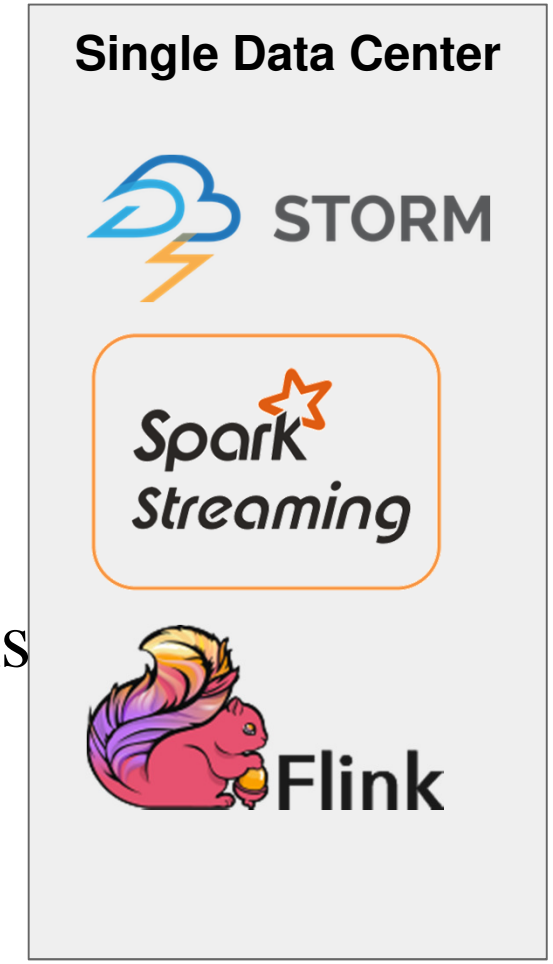
SpanEdge

A multi-data center stream processing solution that provides:

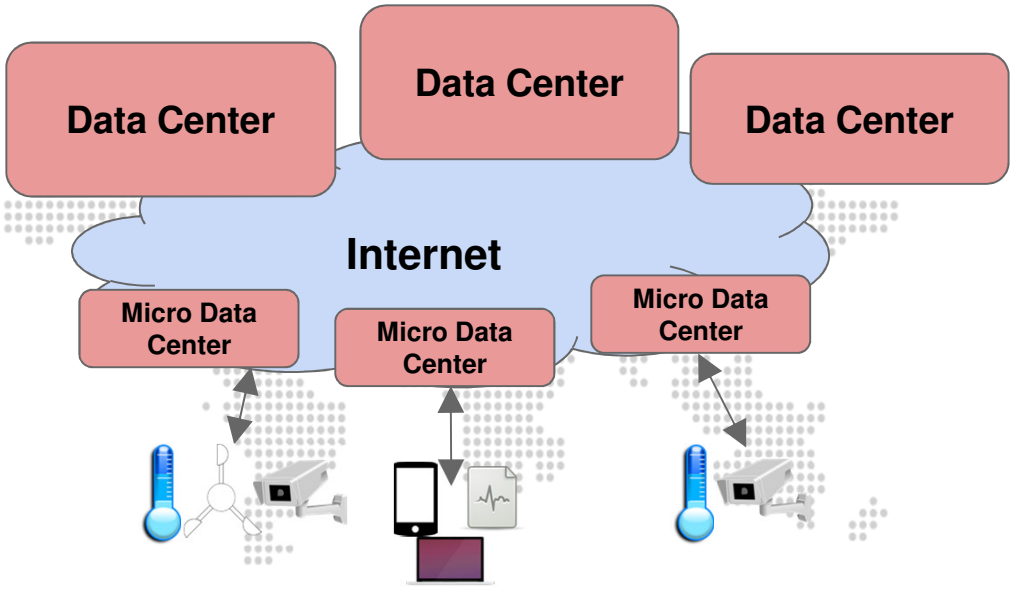
- an expressive programming model to unify programming on a geo-distributed infrastructure.
- a run-time system to manage (schedule and execute) stream processing applications across the DCs.

Stream Processing Systems

- Several open-source stream processing systems
- Run-time system + application development environment
- Multi-applications + multi-streams
- Such as **Apache Storm**, Spark streaming, and Flink



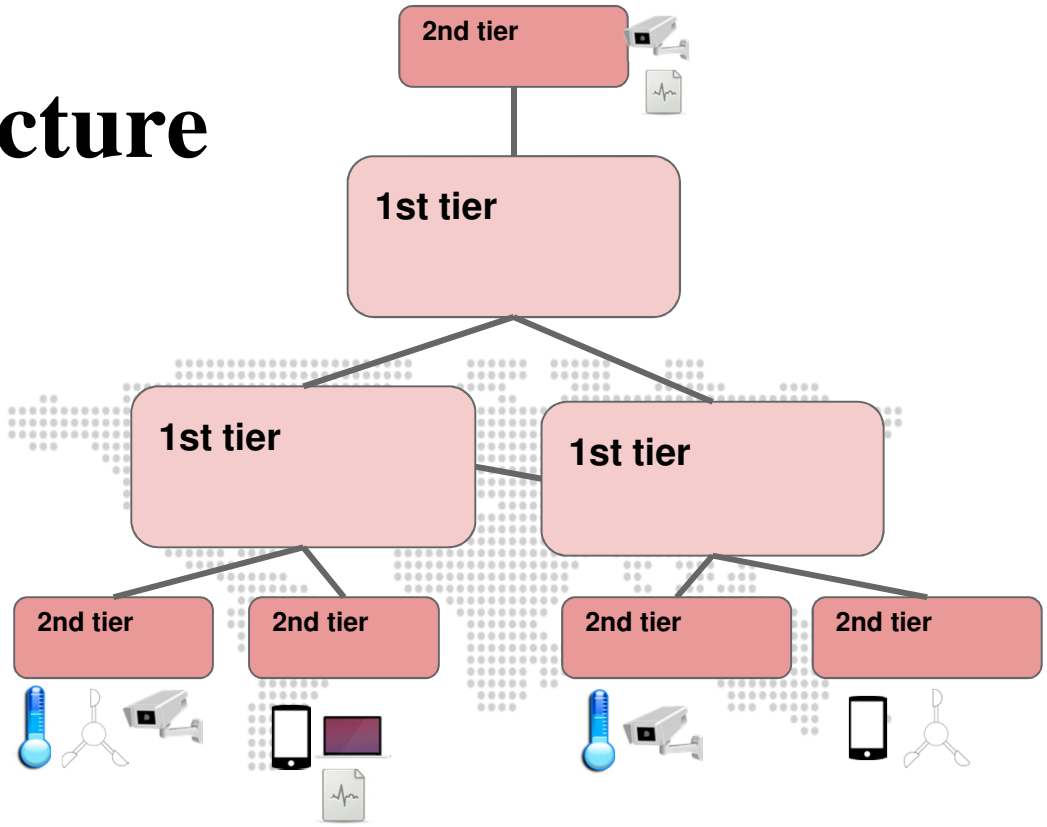
SpanEdge Architecture



SpanEdge Architecture

Two tiers:

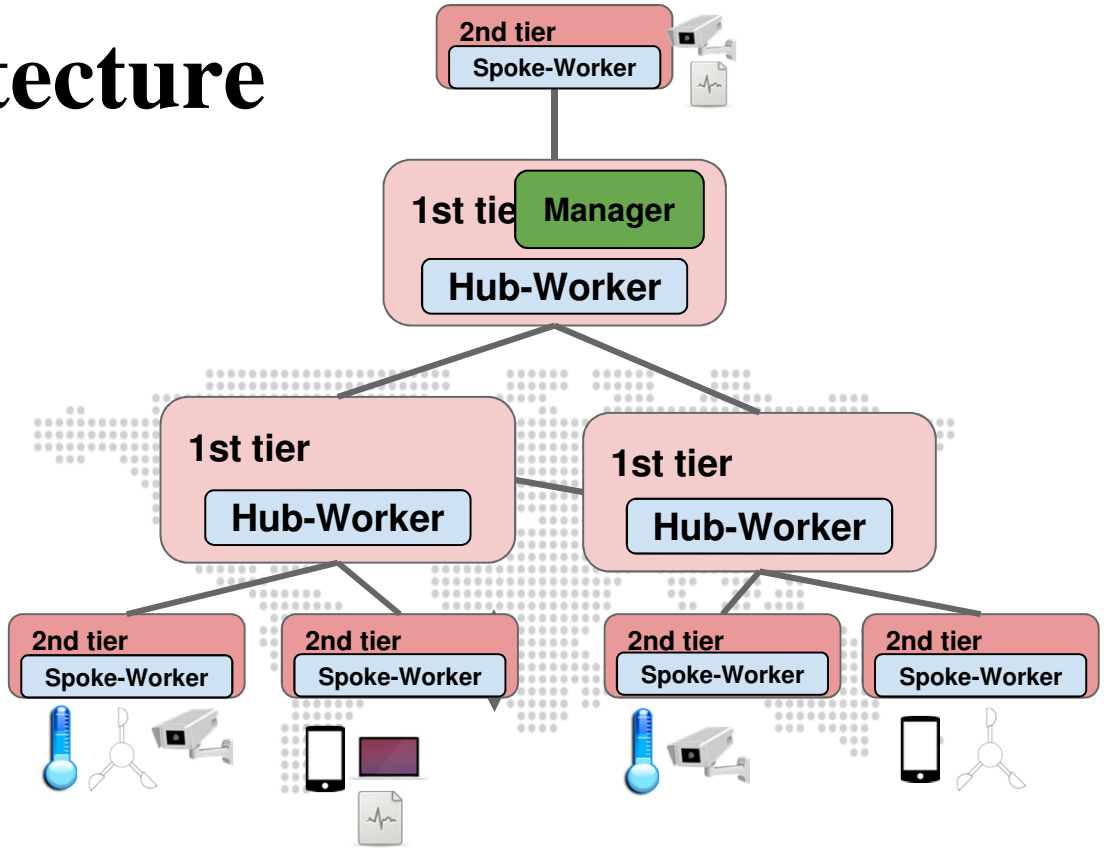
- First tier includes central data centers
- Second tier includes near-the-edge data centers



SpanEdge Architecture

Two types of workers:

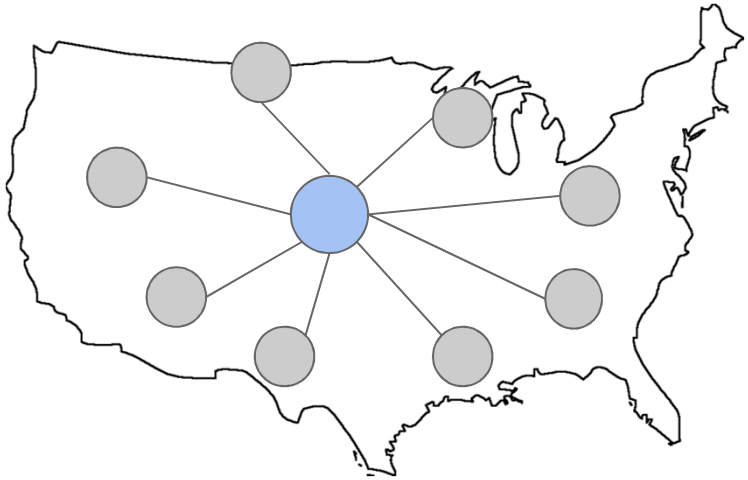
- Hub-worker
- Spoke-worker

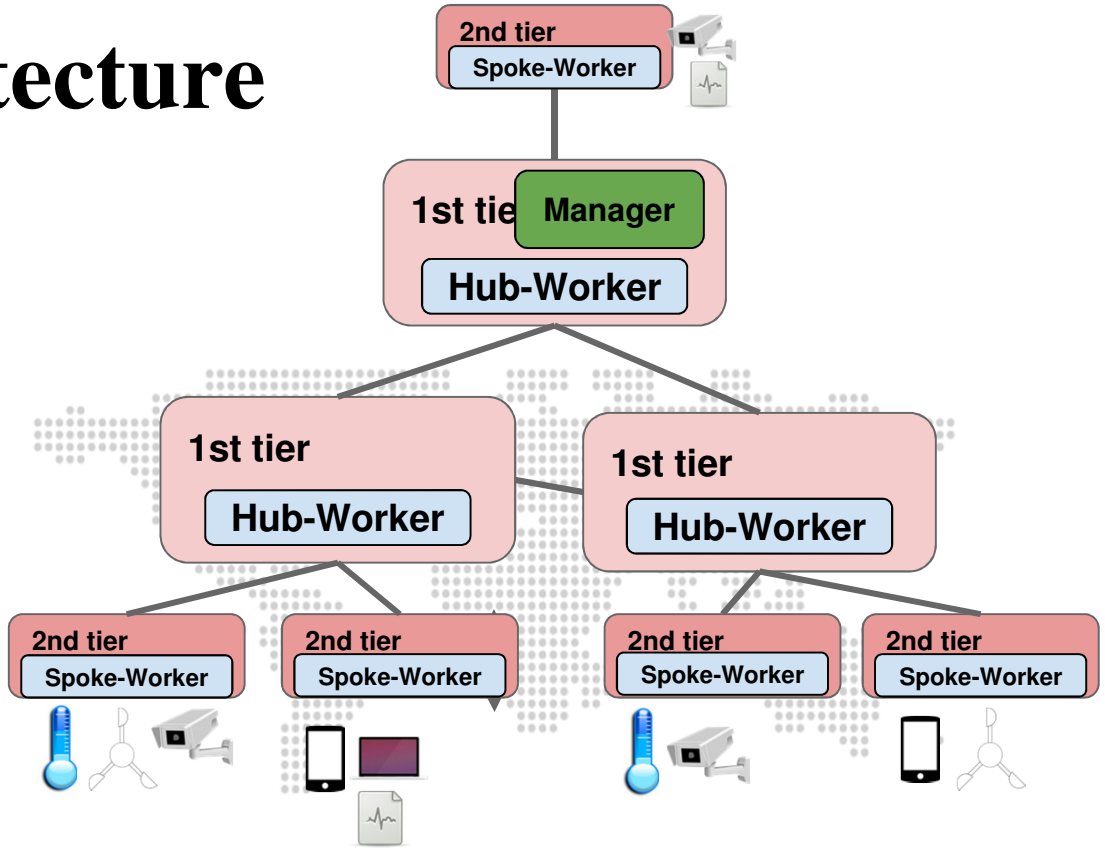


SpanEdge Architecture

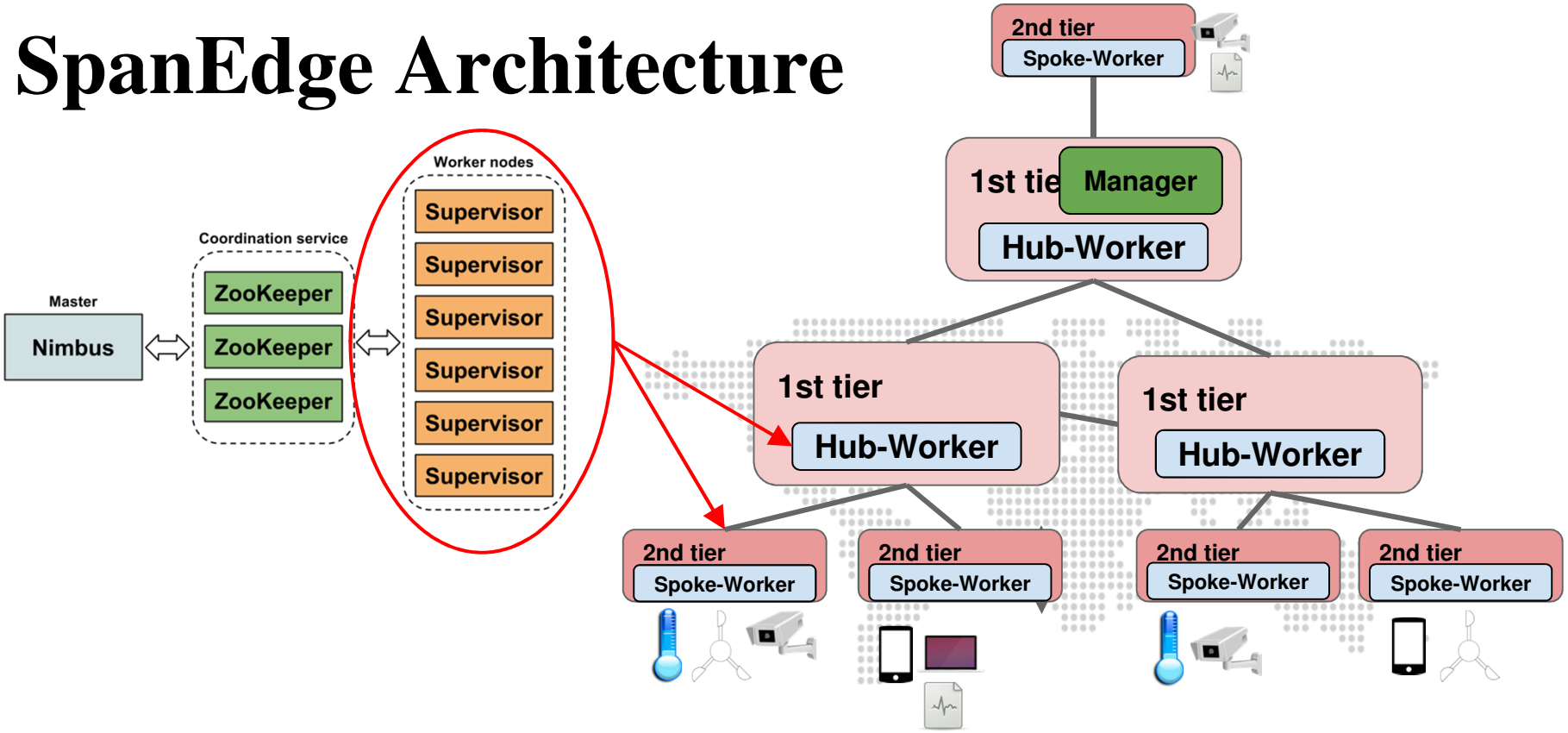
Two types of workers:

- Hub-worker
- Spoke-worker

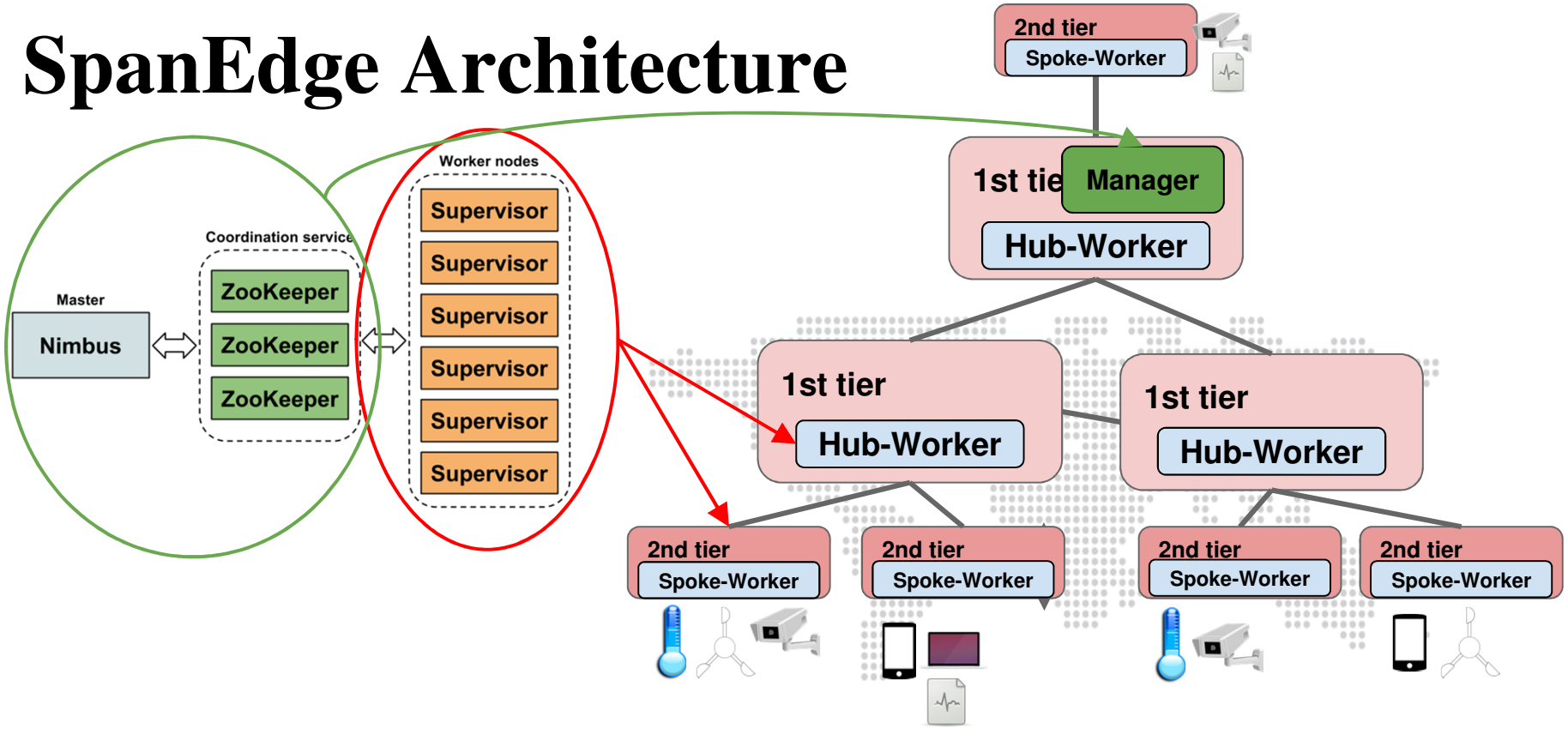




SpanEdge Architecture



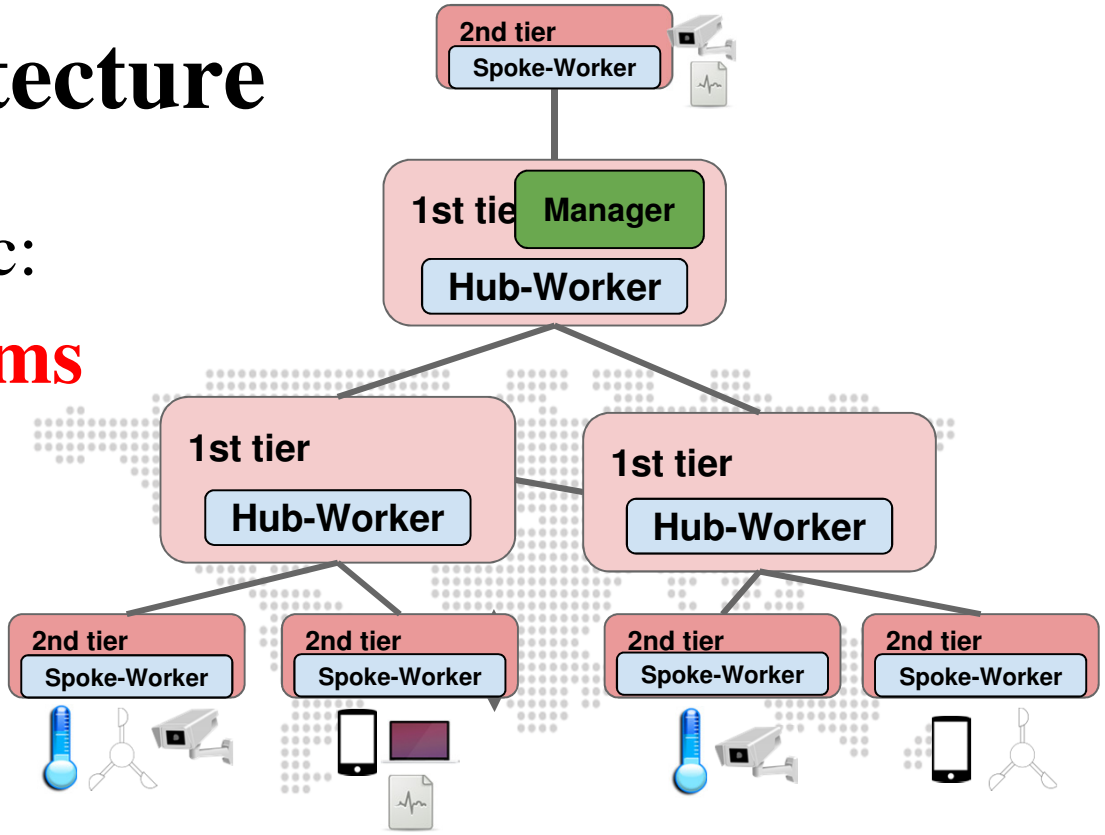
SpanEdge Architecture



SpanEdge Architecture

Cross data center traffic:

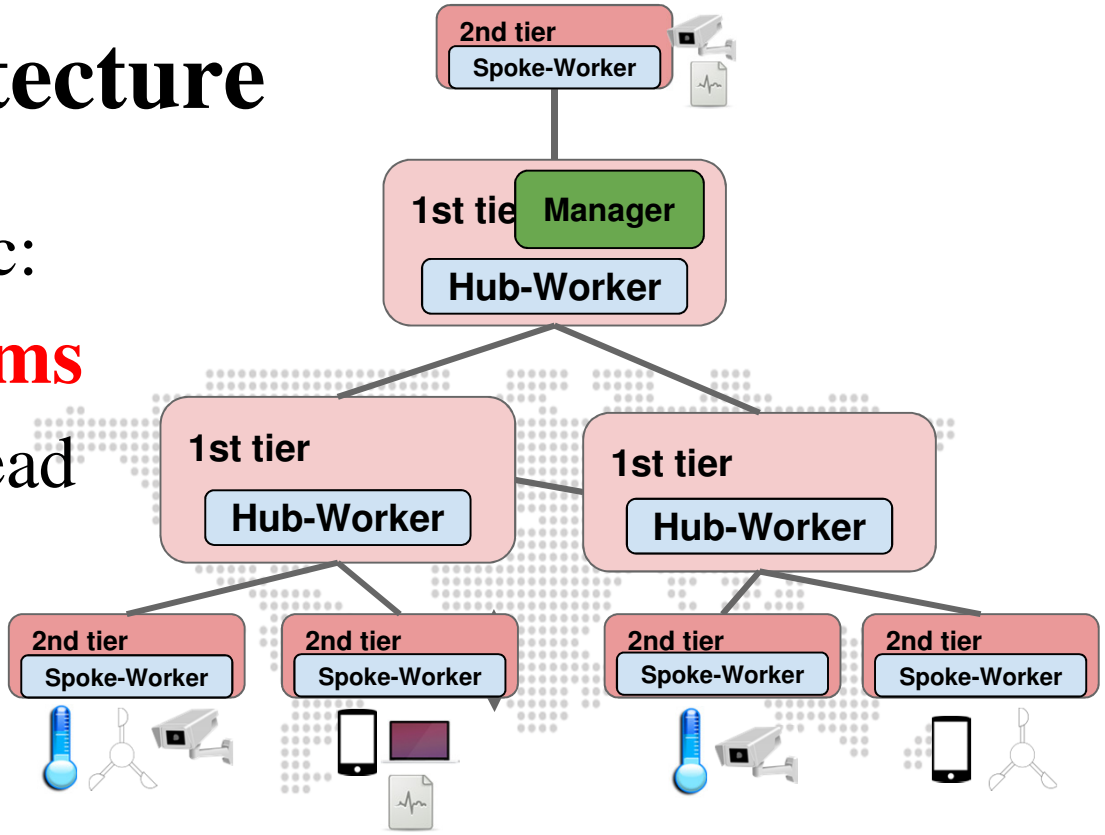
1. **Actual data streams**



SpanEdge Architecture

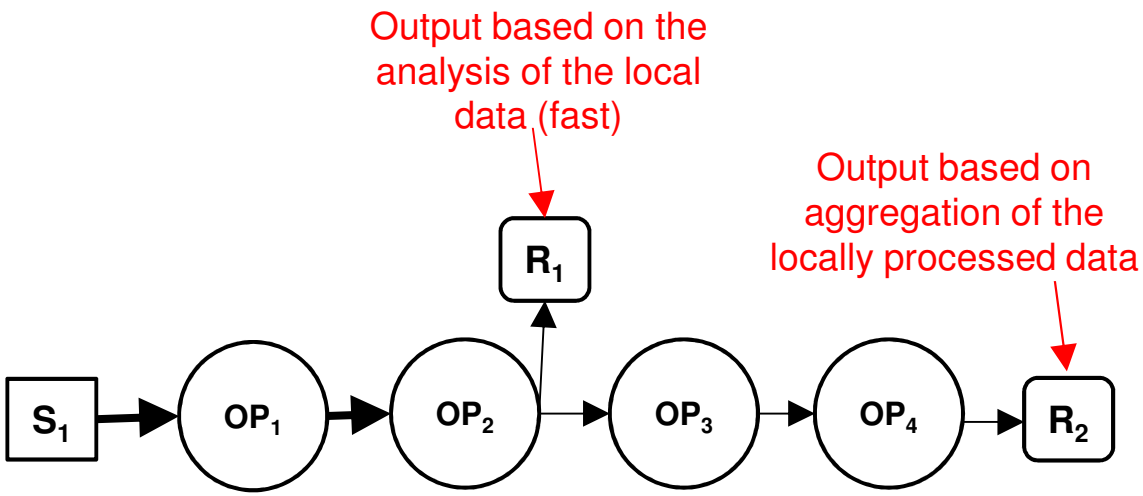
Cross data center traffic:

1. **Actual data streams**
2. Maintenance overhead (workers-manager)



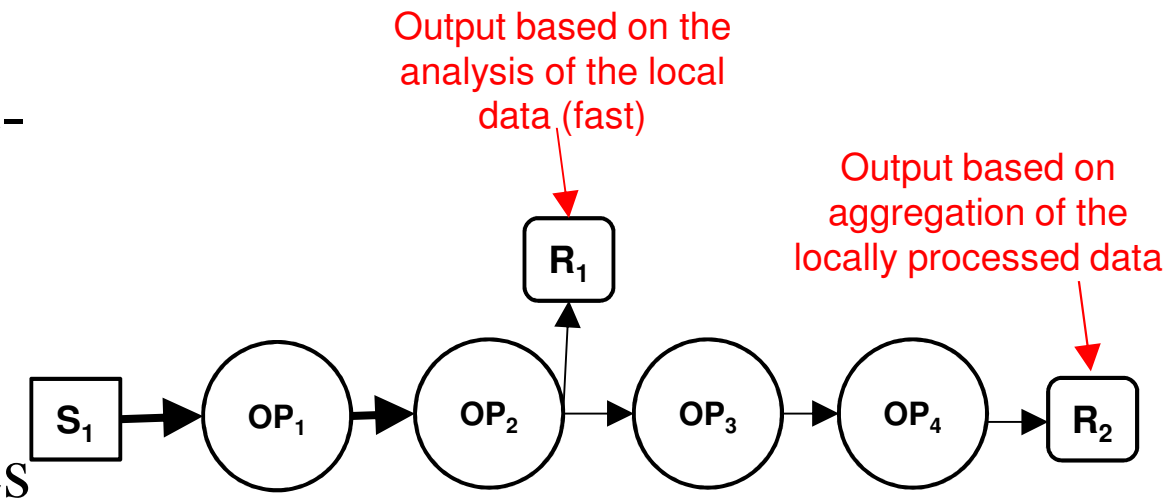
Task Groupings

Task Groupings



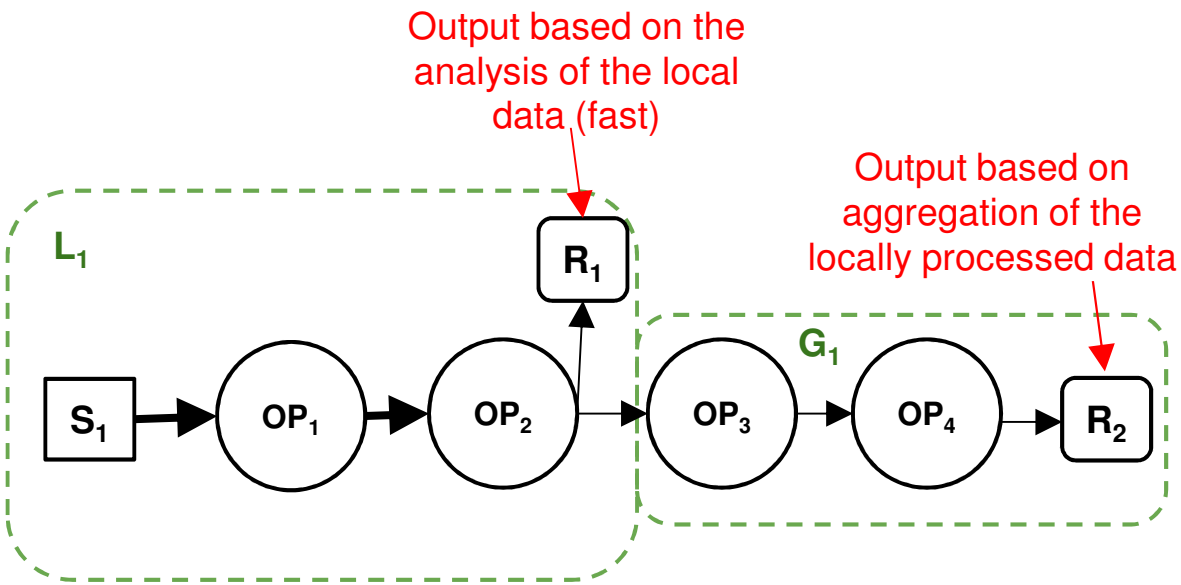
Task Groupings

- Fast results based on the data available near-the-edge
- Avoid sending unnecessary tuples over the WAN



Task Groupings

- **Local-Task:** close to the data source on spoke-workers.
- **Global-Task:** for processing data generated from local-tasks, placed on a hub-worker.



Task Groupings

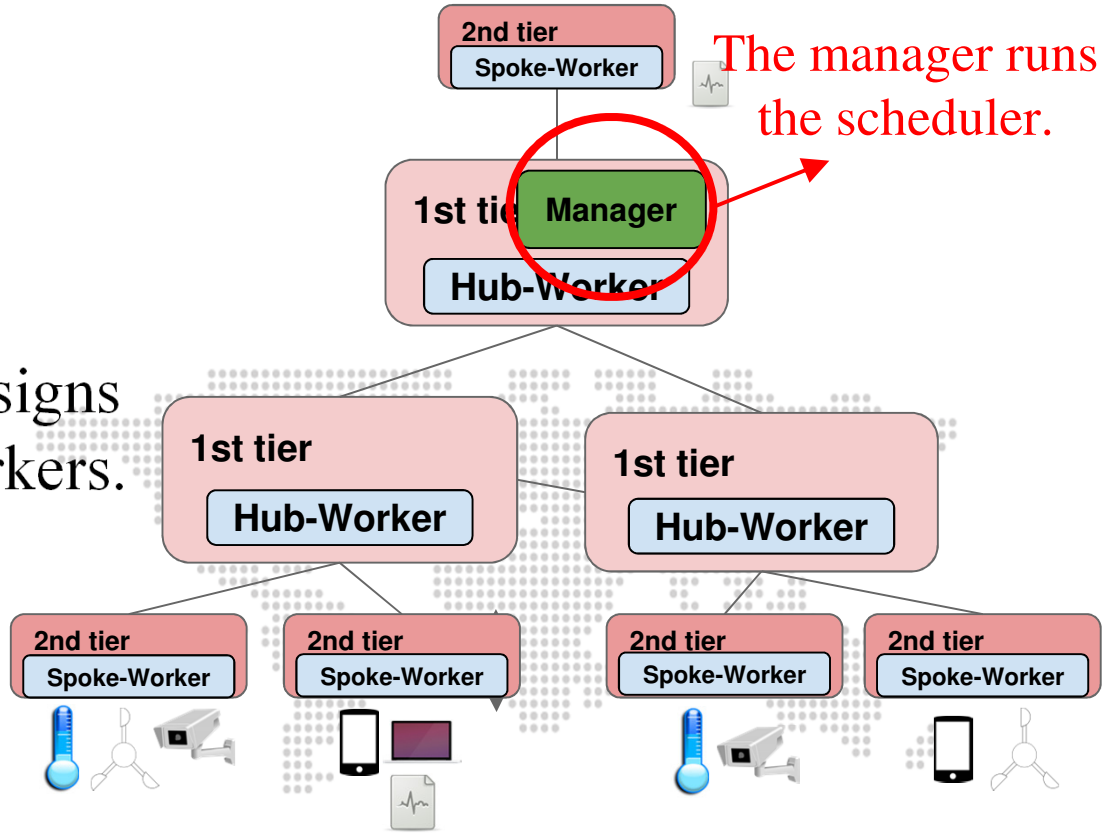
- Defining local-tasks and global-tasks in our implementation:

It can be set as a configuration to TopologyBuilder by the keys local-task and global-task

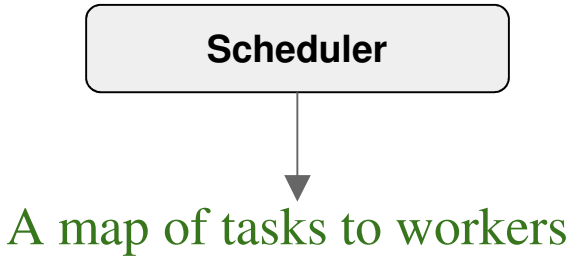
```
TopologyBuilder builder = new TopologyBuilder();
. . .
builder.setSpout("temperatureSpout", tSpout, 4)
    .addConfiguration("local-task", "L1");
...
builder.setBolt("localTempBolt", lBolt, 2)
    .shuffleGrouping("temperatureSpout")
    .addConfiguration("local-task", "L1");
. . .
builder.setBolt("aggregateBolt", aBolt, 4)
    .shuffleGrouping("localTempBolt")
    .addConfiguration("global-task", "G1");
```

Scheduler

- Converts a stream processing graph to an execution graph and assigns the created tasks to workers.

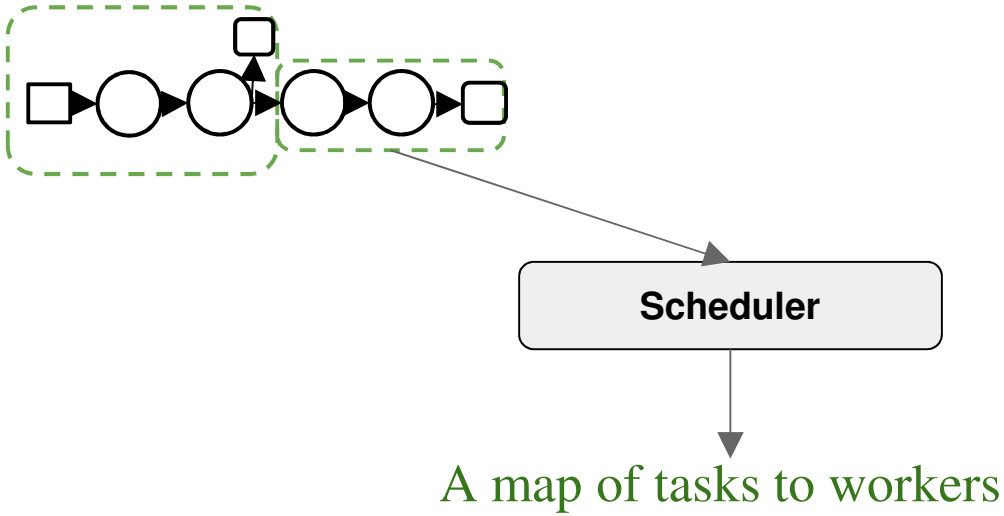


Scheduler



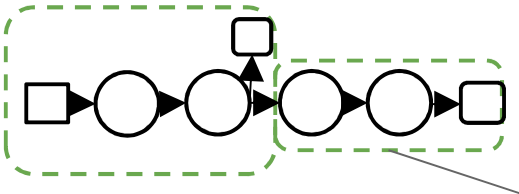
Scheduler

1. A stream processing graph



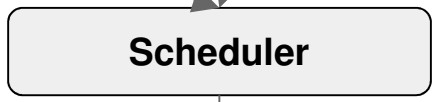
Scheduler

1. A stream processing graph



2. A map of streaming data sources

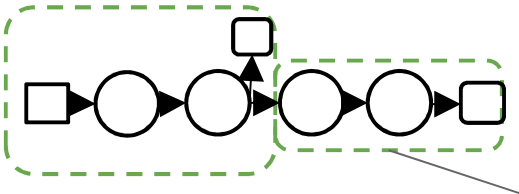
Source Type	Spoke-Worker
src1	{sw1, sw2, sw3}
src2	{sw2,sw4}
….	….



A map of tasks to workers

Scheduler

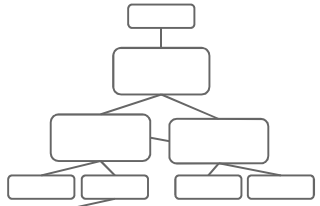
1. A stream processing graph

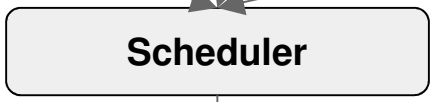


2. A map of streaming data sources

Source Type	Spoke-Worker
src1	{sw1, sw2, sw3}
src2	{sw2,sw4}
....	....

3. The network topology between workers

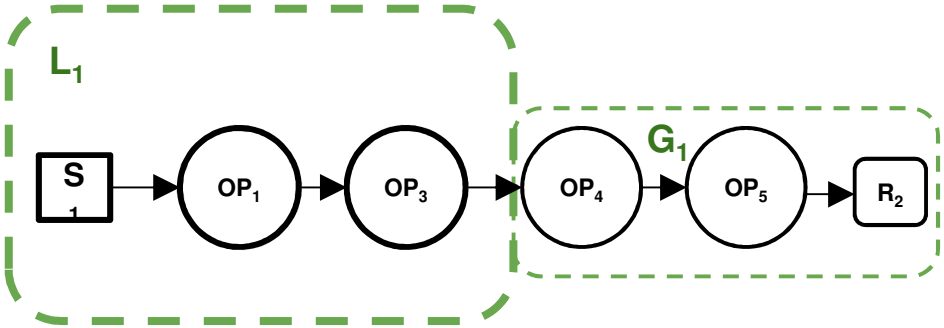


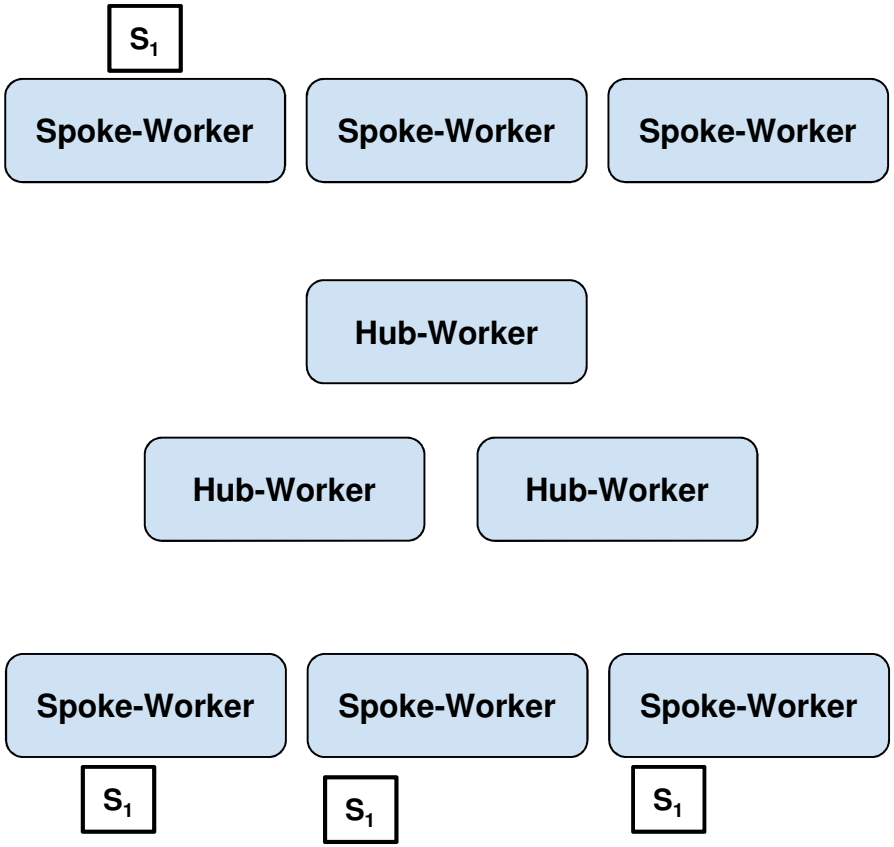


A map of tasks to workers

Scheduler

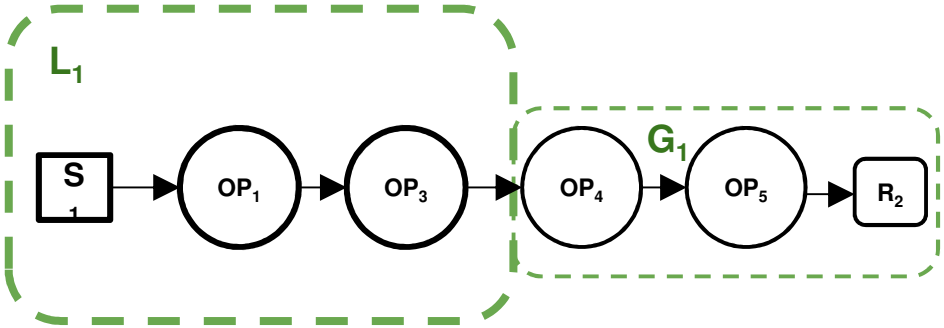
Assigns tasks to workers in two steps:

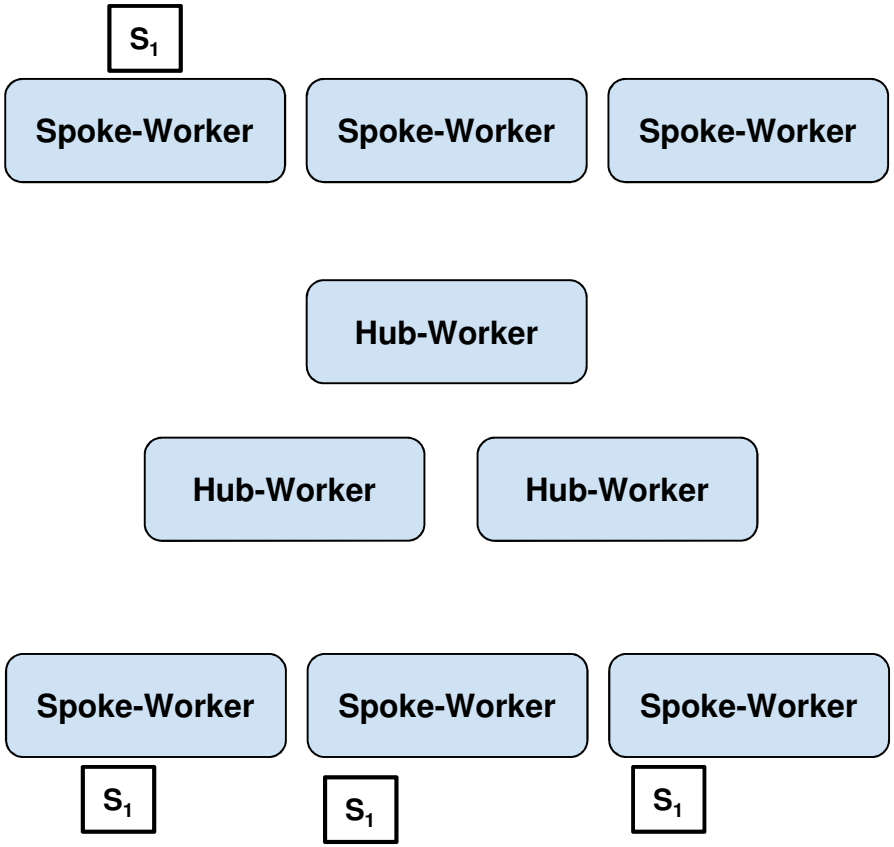




Scheduler

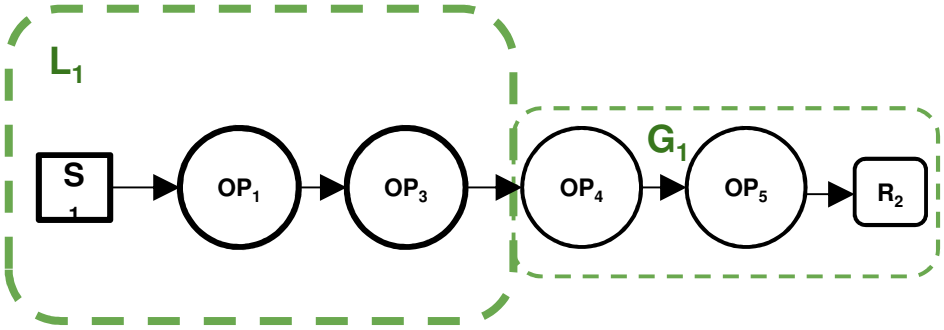
1. assigns local-tasks to spoke-workers.

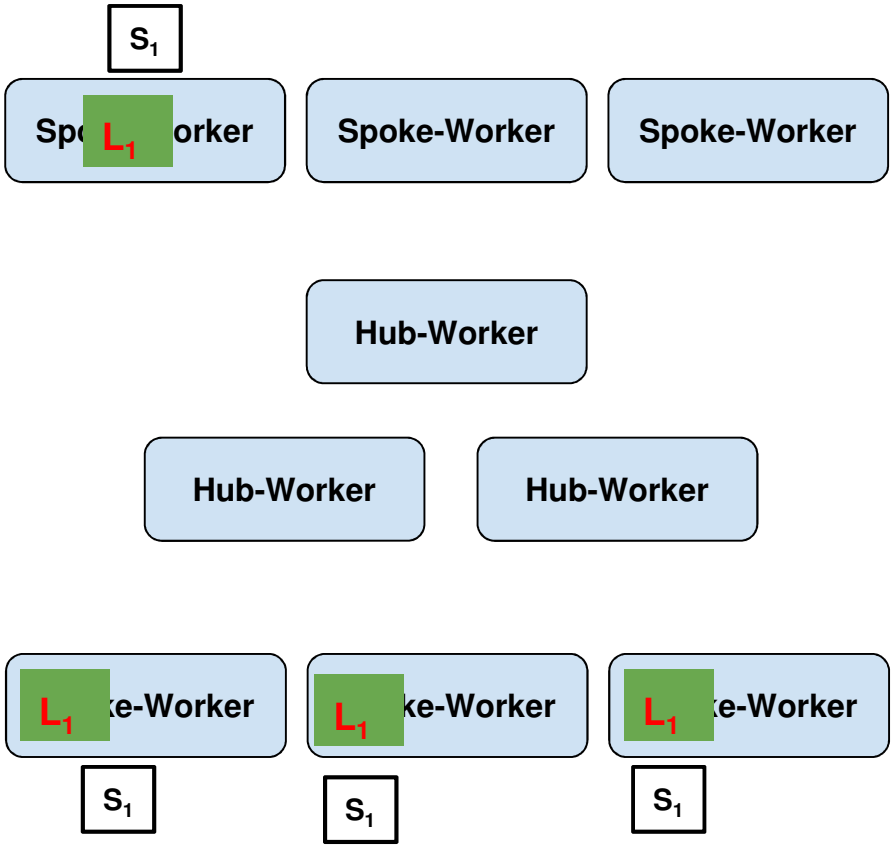




Scheduler

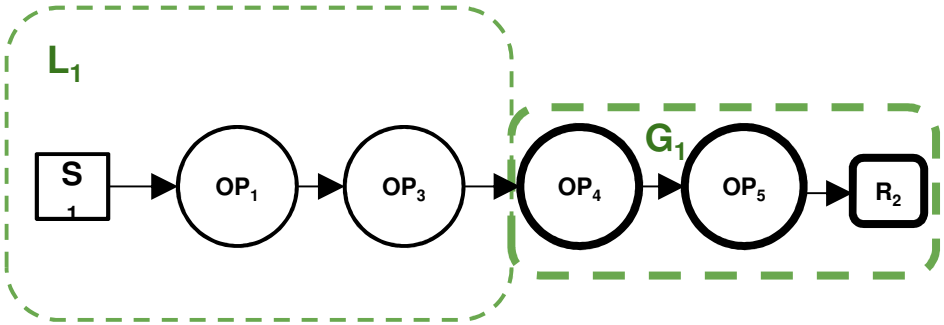
1. assigns local-tasks to spoke-workers.

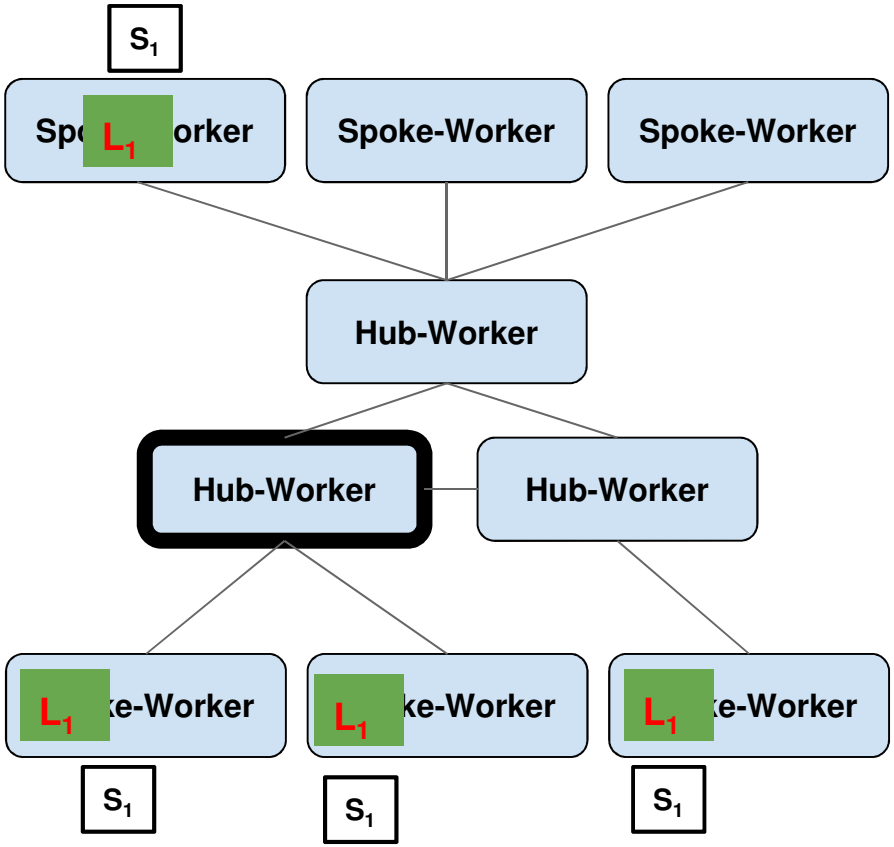




Scheduler

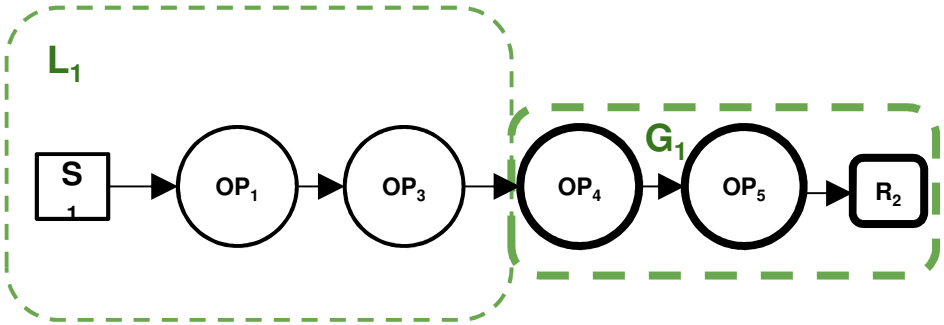
2. assigns global-tasks to the closest hub-worker.

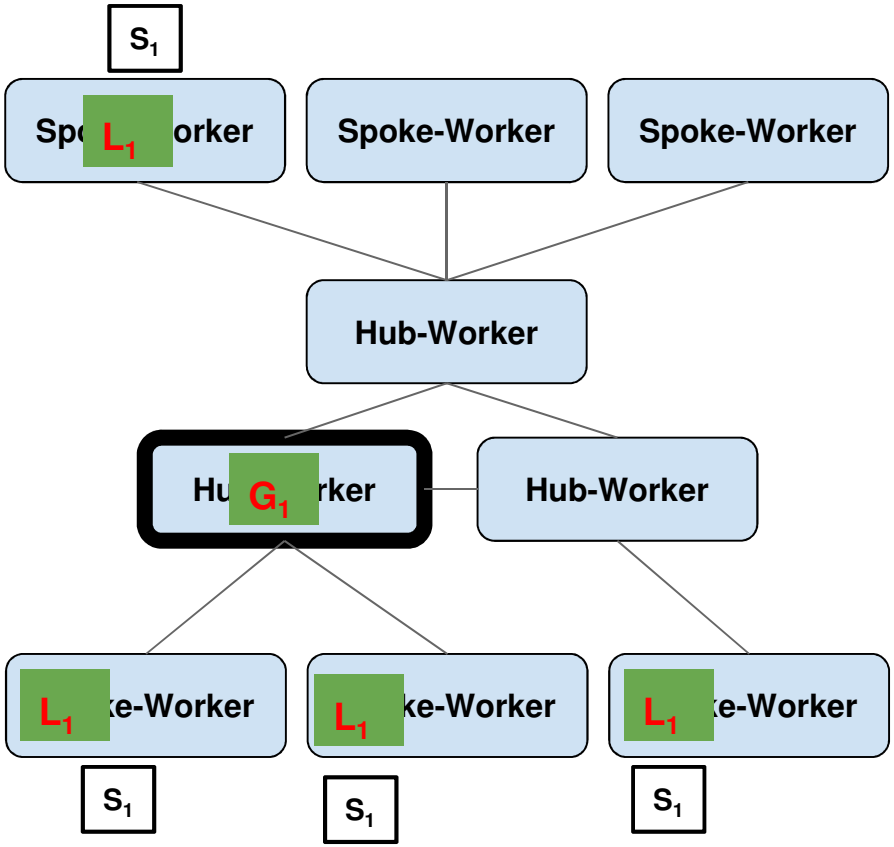




Scheduler

2. assigns global-tasks to the closest hub-worker.





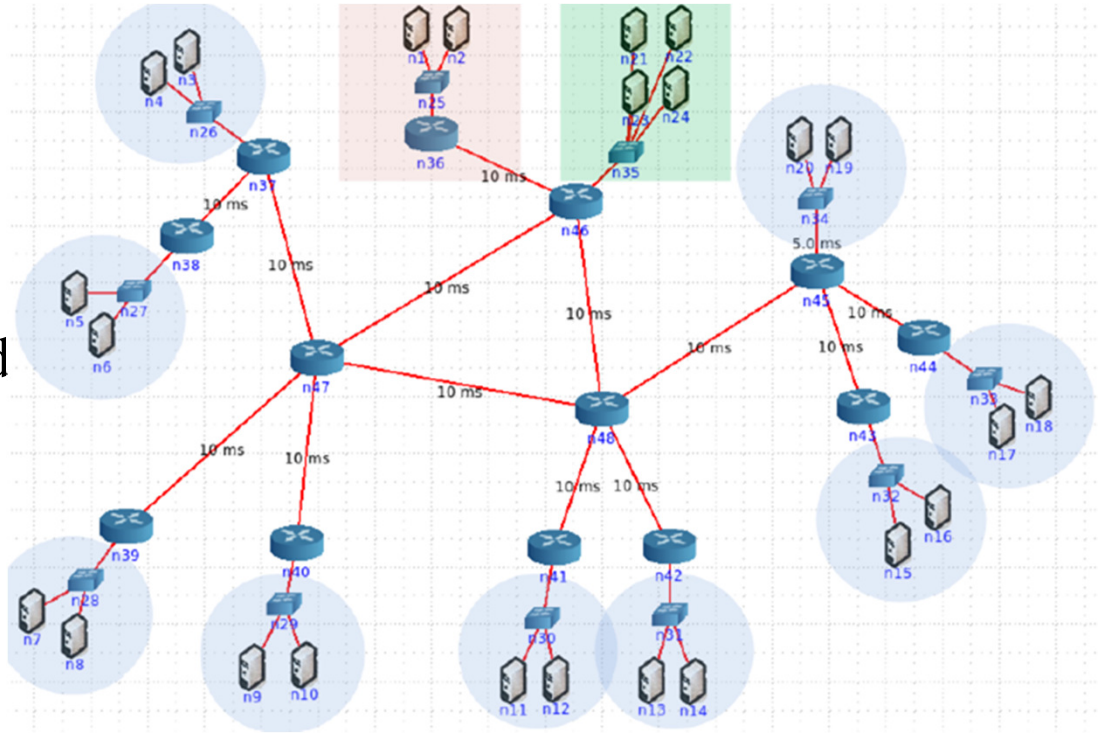
Scheduler: Implementation

- As a plug-in in Apache Storm
- Nimbus (master) executes the scheduler

Evaluation: Infrastructure

- The CORE network emulator
- Our prototype of SpanEdge runs in the Linux containers managed by the CORE emulator
- The manager runs in one of the central data centers

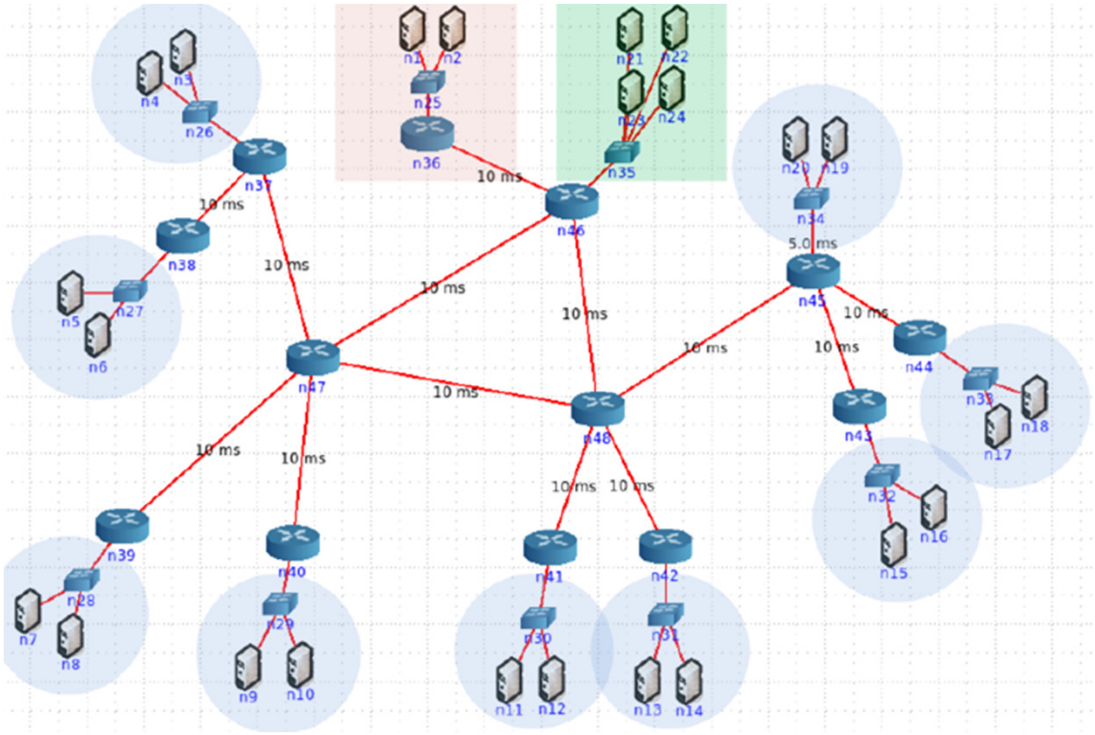
2 central and 9 near-the-edge data centers



Evaluation: Infrastructure

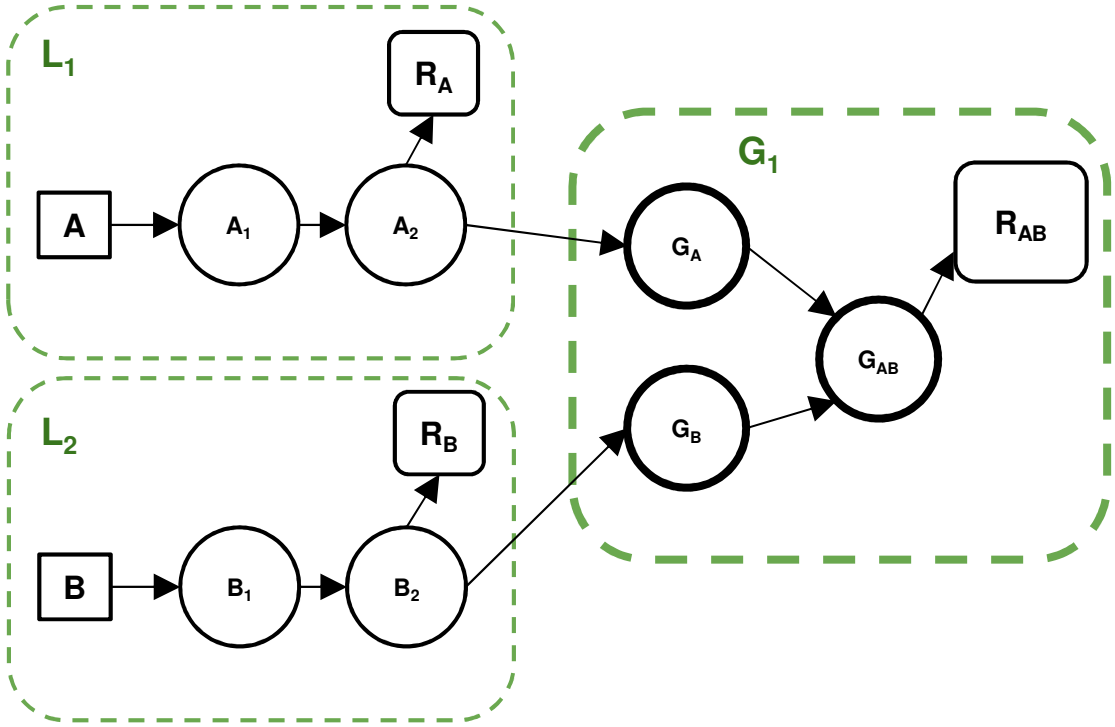
- Compare with the Centralized Approach
- Apache Storm running in one of the central data centers

2 central and 9 near-the-edge data centers

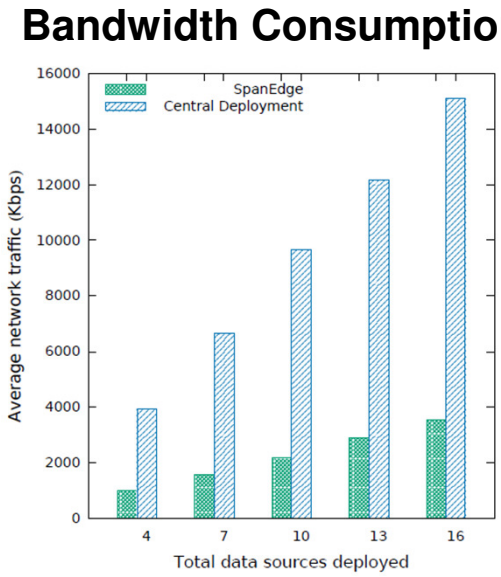


Evaluation: Stream Processing Graph

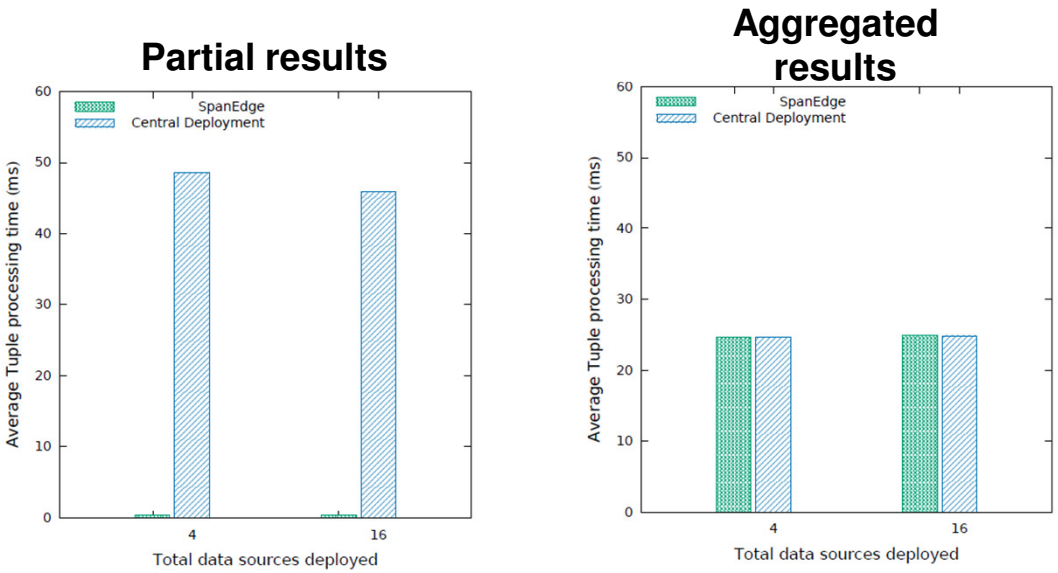
- 2 stream sources:
  Type A and
  Type B



Evaluation: Bandwidth



Evaluation: Latency



Conclusions

SpanEdge:

- facilitates programming on a geo-distributed infrastructure including central and near-the-edge data centers
- provides a run-time system to manage stream processing applications across the DCs.

Future Work

- A dynamic scheduler
- Mobility of the data sources and their state migration
- Fault-tolerance mechanisms in geo-distributed infrastructure

Thank You!

The source code is available at:
www.github.com/telolets/stormonedge