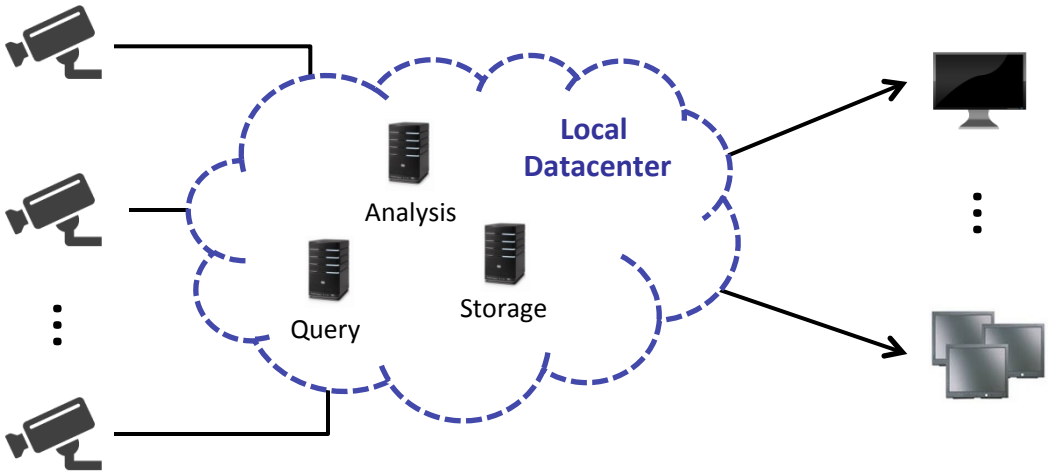
Processing of Mix-Sensitivity Video Surveillance Streams on Hybrid Clouds

Chunwang Zhang, *Ee-Chien Chang*
School of Computing, National University of Singapore
28th June, 2014

Outline

1. **Motivation**
2. Hybrid Cloud Video Surveillance Model
3. Scheduler
4. Evaluation
5. Conclusions

1. Motivation



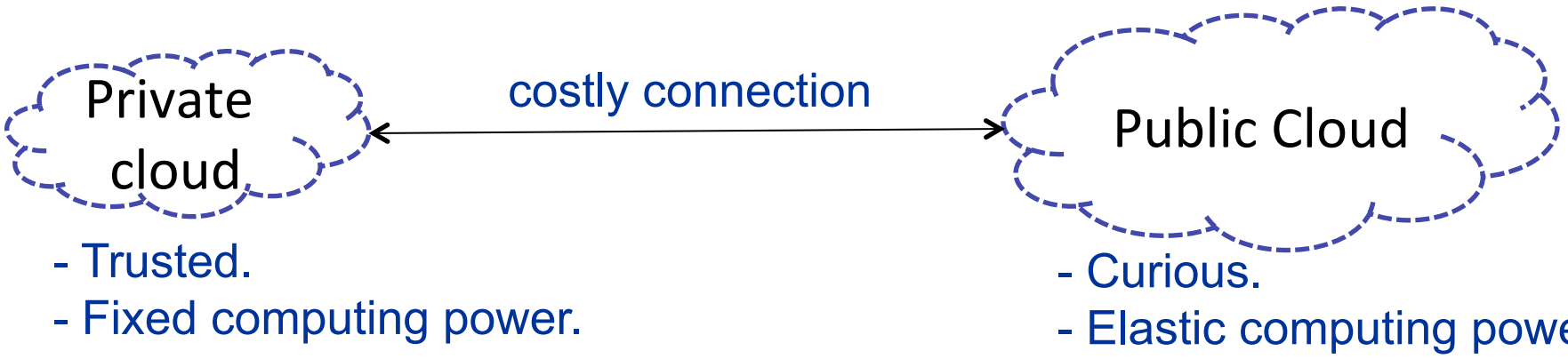
Video Surveillance Systems

- **Video surveillance systems are inherently data-intensive and often compute-intensive**
  - Transcoding, indexing, video analytics etc
  - Workload could be seasonal

1. Motivation

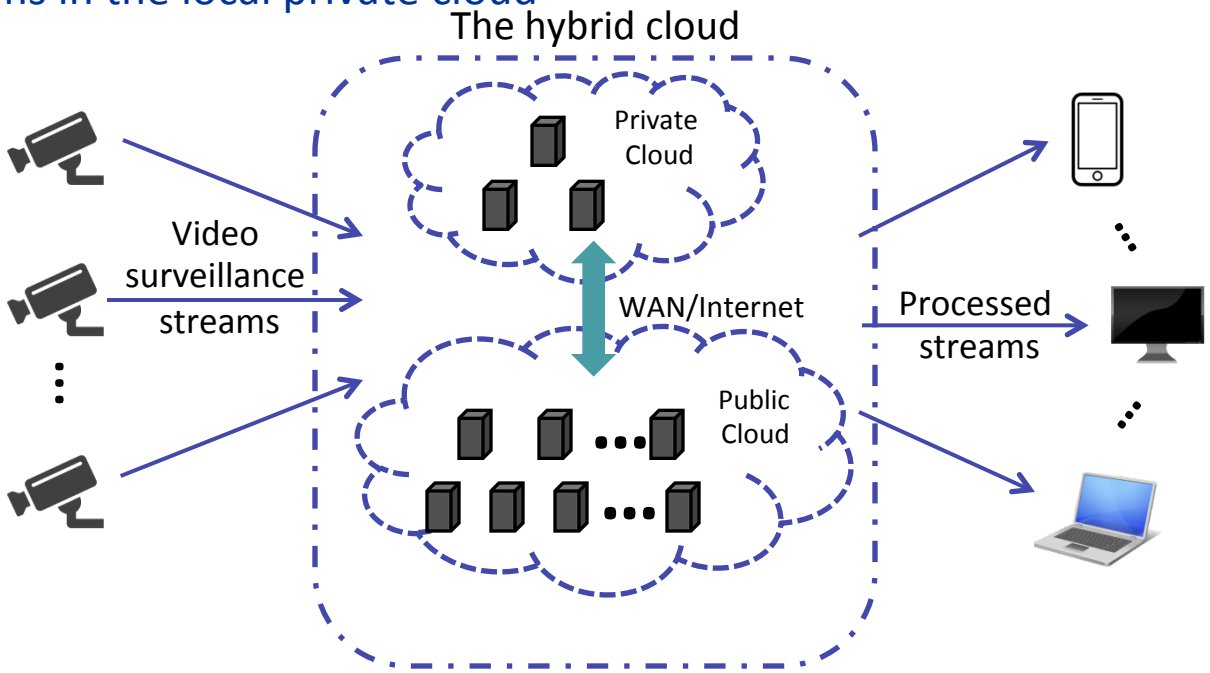
- **Outsourcing to public cloud (e.g., Amazon AWS)?**
  - Surveillance videos could contain sensitive info.
    - Various data breaches were report for different cloud provider
  - Processing in the encrypted domain is too costly for large video data.

- **We consider the approach of data/computations segregation in the hybrid cloud.**



1. Motivation

- **A hybrid cloud-based video surveillance system**
  - Pushing partial video streams to public cloud while keeping sensitive video streams in the local private cloud



- **It's desired to have a middleware that unifies the two clouds and schedules the computation effectively.**

1. Motivation

- **Previous works on distributed stream processing focus on scheduling among multiple servers to balance workload among all the servers, etc.**

- **Our problem can be treated as a special case of some known general scheduling models but has its difference**
  - Conceptually consists of only two servers – a private and a public server

Our Works

- **A stream processing model specifically designed for the hybrid cloud setting**
  - Can handle ad-hoc queries and dynamic clients without rescheduling
- **Formulation of the scheduling problem**
  - Minimizes the monetary cost to be incurred on public cloud, subject to resources, security and QoS constraints
- **An efficient scheduler**
- **A proof-of-concept system**

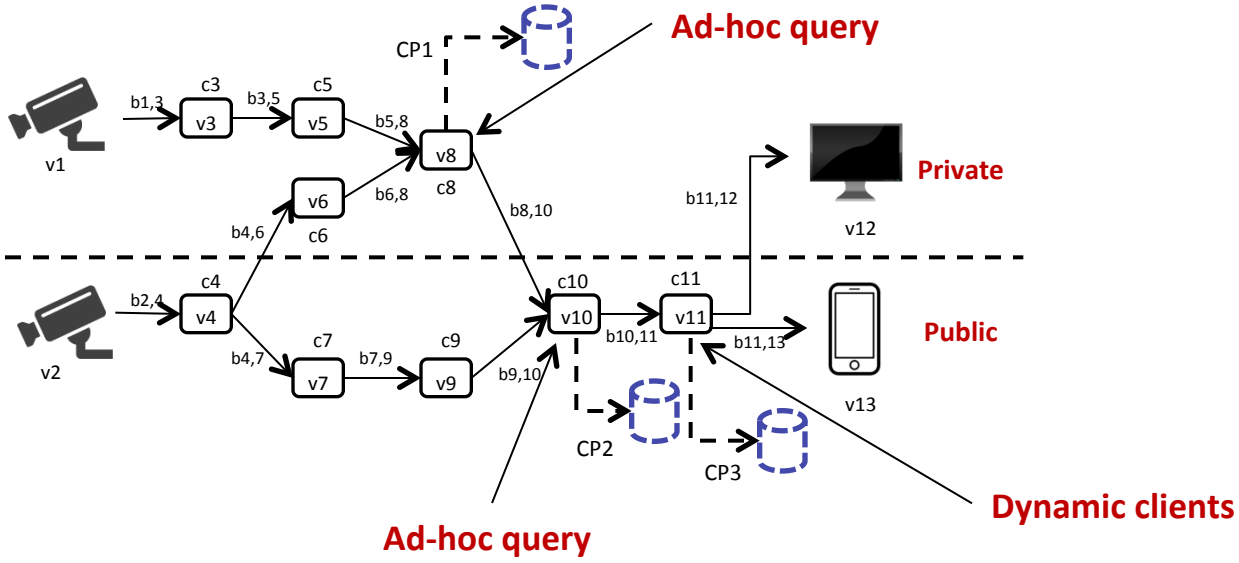
Outline

1. Motivation
2. **Hybrid Cloud Video Surveillance Model**
3. Proposed Scheduler
4. Evaluation
5. Conclusions

2. Hybrid Cloud Video Surveillance Model

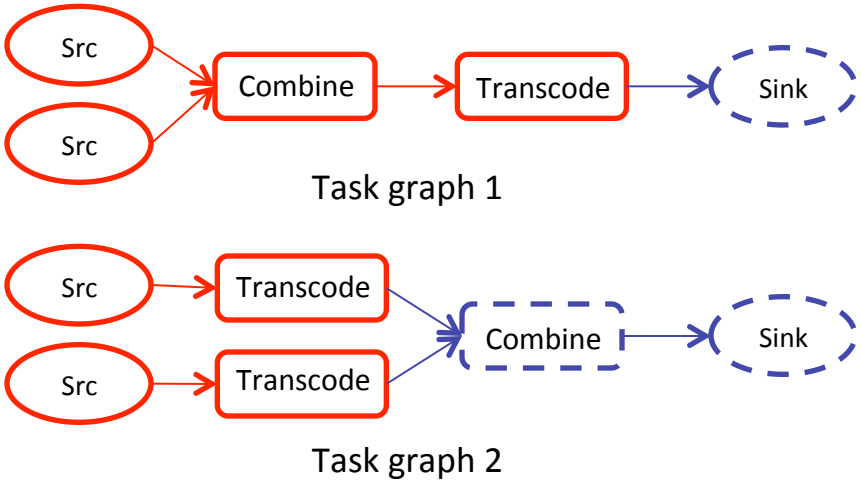
Stream Processing Model

- Each task template is modeled as a directed, acyclic and labeled graph.
  - Each template can be instantiated to multiple sources/sinks.
  - Connection Points (CPs): where ad-hoc queries and "dynamic clients" can attach

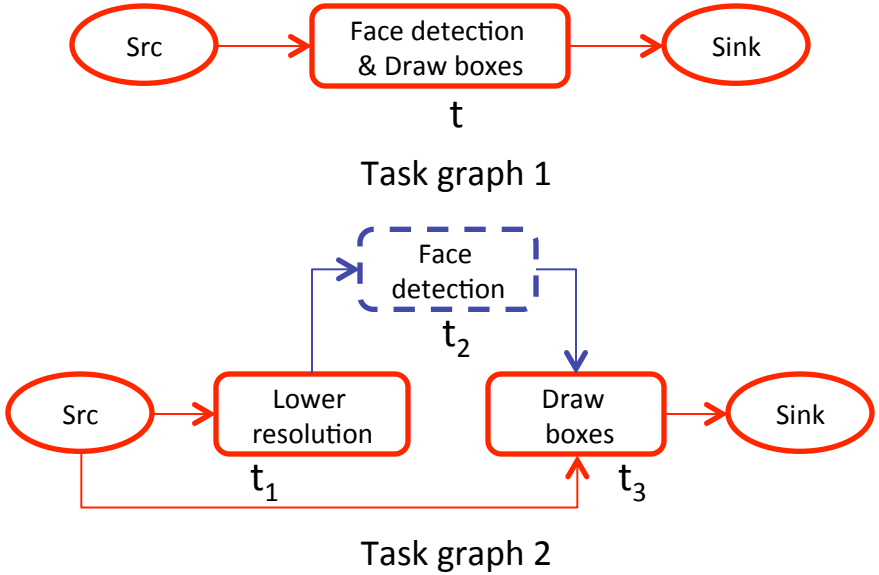


2. Hybrid Cloud Video Surveillance Model

Some tasks could be completed in multiple ways:



Example 1

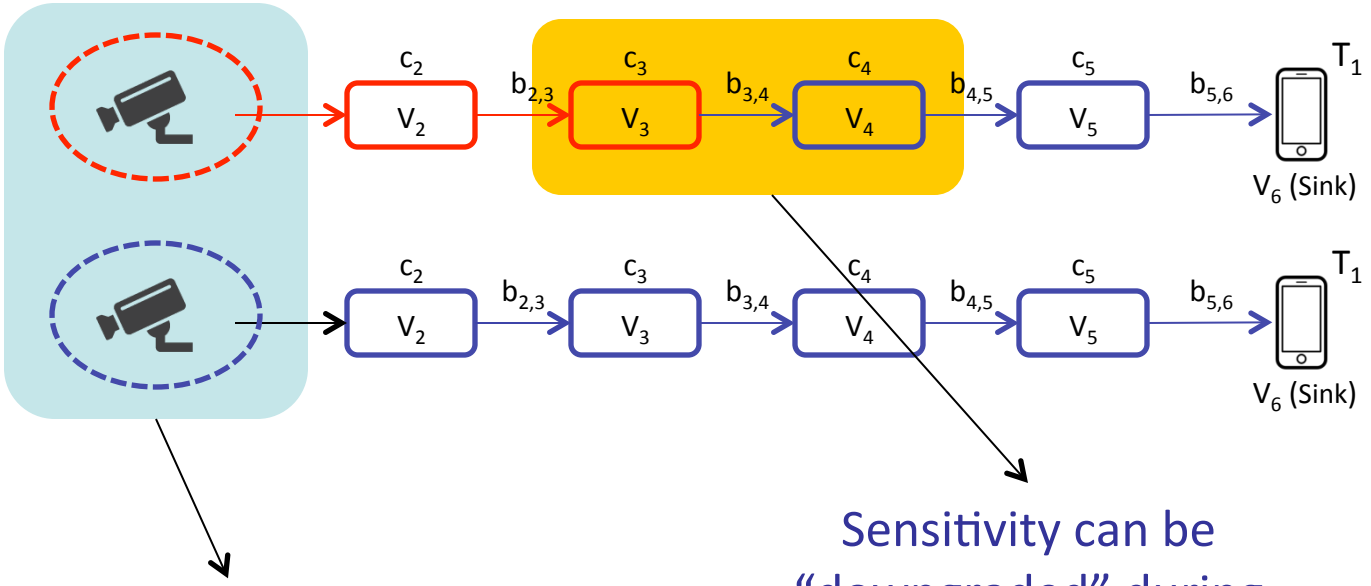


Example 2

2. Hybrid Cloud Video Surveillance Model

Security Model

- Each node in an instantiated task is tagged as *sensitive* or *non-sensitive*

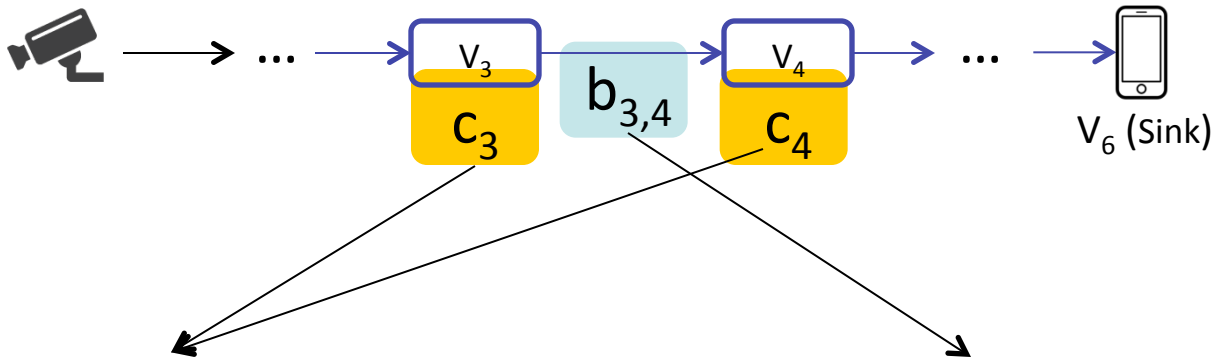


"Videos generated by cameras in the meeting room is sensitive iff time is between 2--4pm"

2. Hybrid Cloud Video Surveillance Model

Cost Model

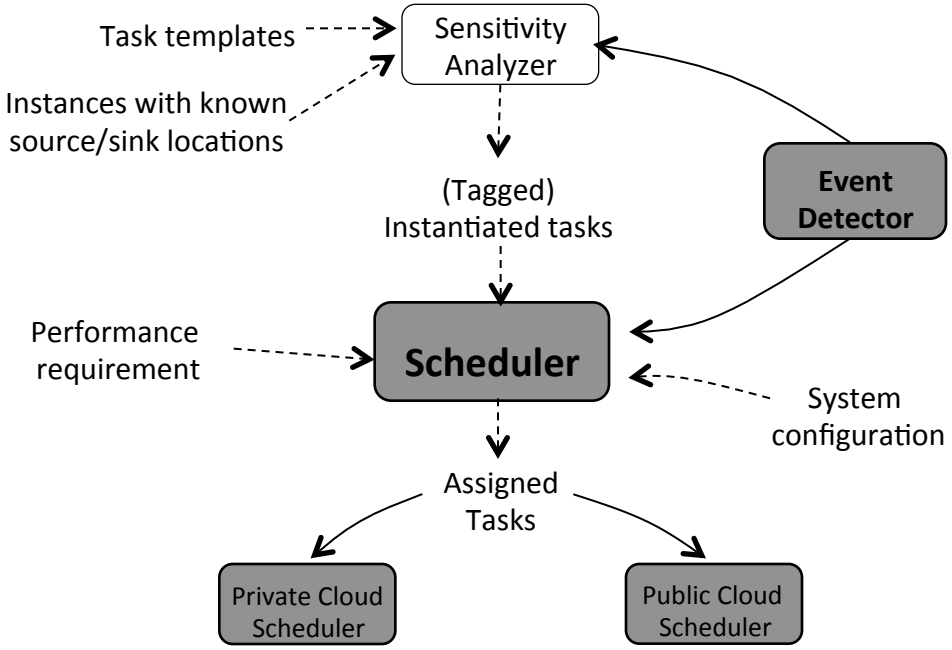
- Approximates actual monetary cost to be incurred



*Each ECU provides the equivalent CPU capacity of a 1.0–1.2 GHz 2007 Opteron or Xeon processor.

2. Hybrid Cloud Video Surveillance Model

System Architecture



Outline

1. Motivation
2. Hybrid Cloud Video Surveillance Model
3. Scheduler
4. Evaluation
5. Conclusions

3. Scheduler – problem formulation

- **Given:** a set of task templates, and the number of time each template is to be instantiated.

- **Find:** The assignment of every operations in each instantiated task such that the cost incurred is minimized, subject to:

  (1) Private cloud cannot be overloaded;

  (2) Sensitive streams cannot flow into public cloud;

  (3) Delay constraint for each assigned task can be met.

3. Scheduler – problem formulation

Cost function

parameter α and β can be determined by the cloud pricing model in use, e.g., α = 0.08 and β = 0.684 according to Amazon

$$COST = \alpha \sum_{i,j} c_j^i (1 - x_j^i) + \beta \sum_{i,j,k} b_{j,k}^i |x_j^i - x_k^i|$$

Computation cost (α * number of ECUs)

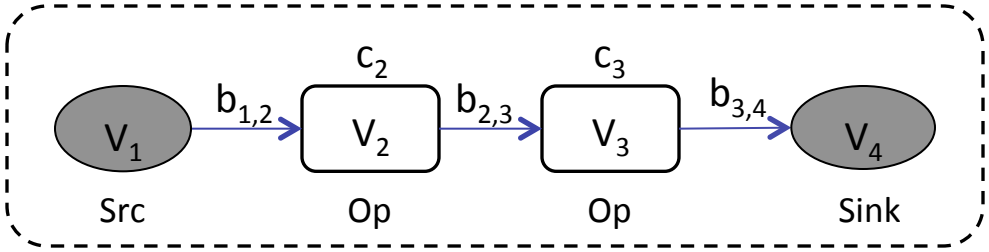
communications cost (β * inter-cloud bandwidth)

x_j^i: assignment of operation *j* in instantiated task *i*.
Value 0 if assigned to public cloud; 1 otherwise.

3. Scheduler

- **Not surprisingly, finding the optimal solution is NP-hard.**

- **Our solution:**
  - Reduce the states space to a smaller set of "minimal configurations", and then employs integer programming to select the desired configurations



in this example, there
are 4 possible configurations:
whether V_2, V_3 to be in public or private.
Not all configurations need to be considered

  - In cases where the problem size are still too large for the integer programming solver, employ a heuristic to further reduce the number of configurations.

Outline

1. Motivation
2. Hybrid Cloud Video Surveillance Model
3. Scheduler
4. Evaluation
5. Conclusions

4. Evaluation

Conducted two groups of experiments:

- **Large-scale simulations**
- **Proof-of-concept system evaluation on Amazon EC2**

We consider 5 schedulers

1) Task-Level Water-filling (TLW): if a task contains sensitive operations, the whole task is assign private, except for sinks. Otherwise, assign it to public.

2) Task-Level Random (TLR): same as TLW but randomly assign non-sensitive task.

3) Greedy: Consider the task one by one, using the optimal assignment for each of them.

4) ProposedC: our scheduler with objective to minimize monetary cost

5) ProposedB: our scheduler with objective to minimize bandwidth usage

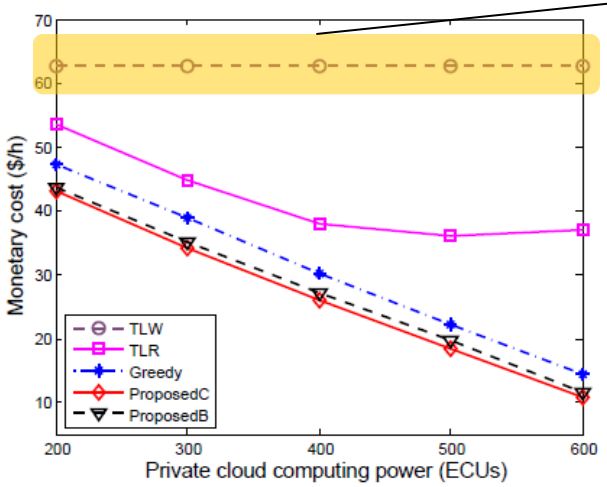
4. Evaluation: simulations

- **Simulation Setting**
  - 10 different task templates where the number of operations nodes ranging from 3 to 15
  - Choose compute cost in (0,2] ECUs and bandwidth cost in (0,1] MB/s
  - Each template is to be instantiated to 10 streams. Randomly set the sensitivity.
  - Private cloud ranges from 200 to 600 ECUs, Delay constrain: 250ms

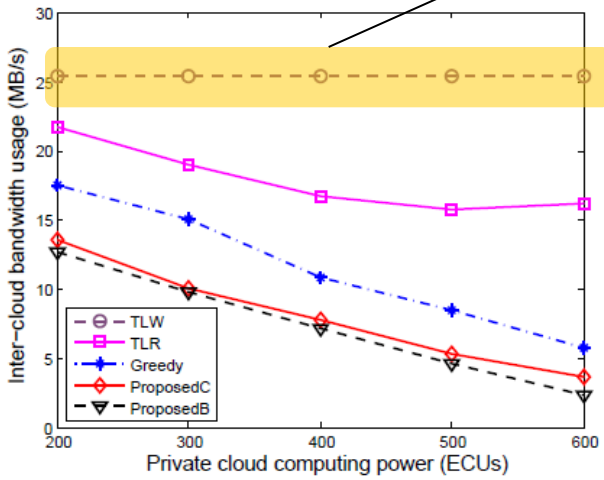
4. Evaluation: simulations

- **Without security constraint**

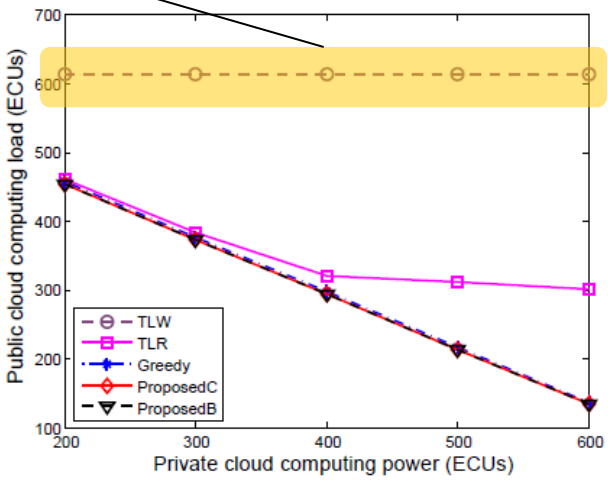
TLW pushes all streams to public cloud, giving highest monetary cost and bandwidth usage



(a) Monetary cost



(b) Inter-cloud bandwidth

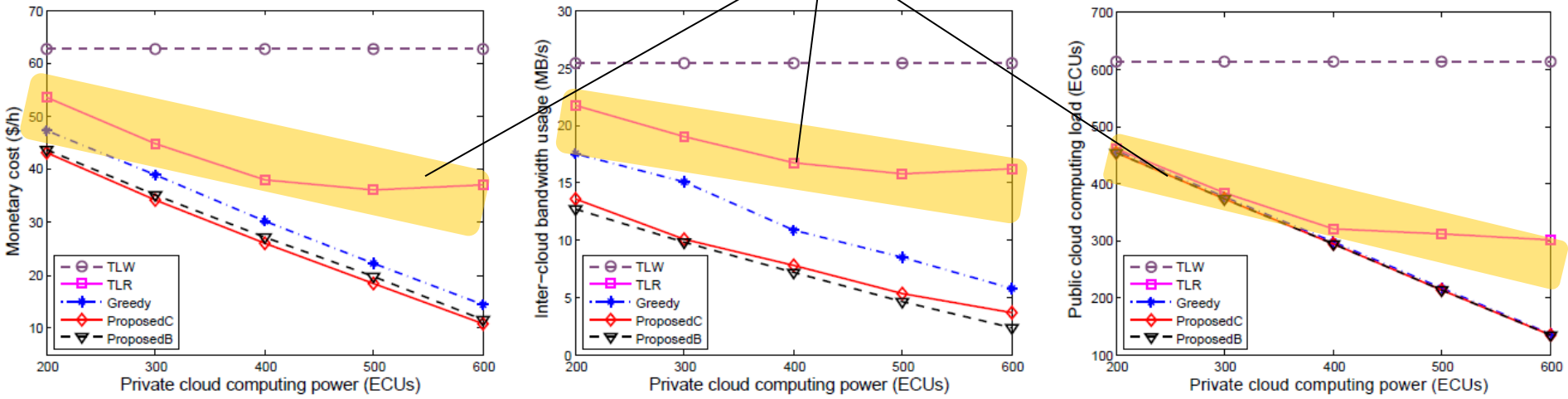


(c) Outsourced computation

4. Evaluation: simulations

- **Without security constraint**

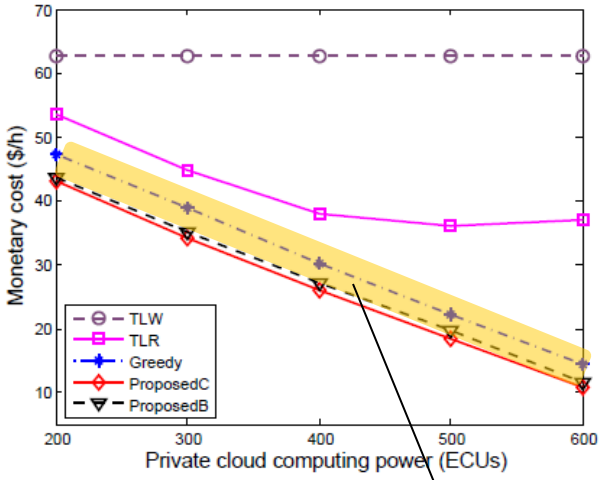
TLR keeps some streams locally but still underutilizes the private cloud



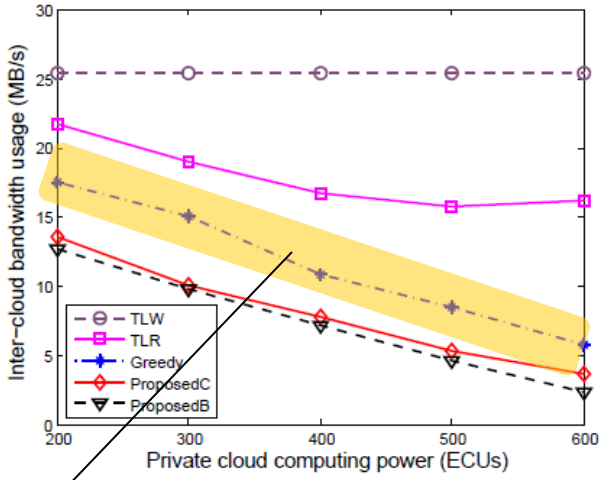
(a) Monetary cost (b) Inter-cloud bandwidth (c) Outsourced computation

4. Evaluation: simulations

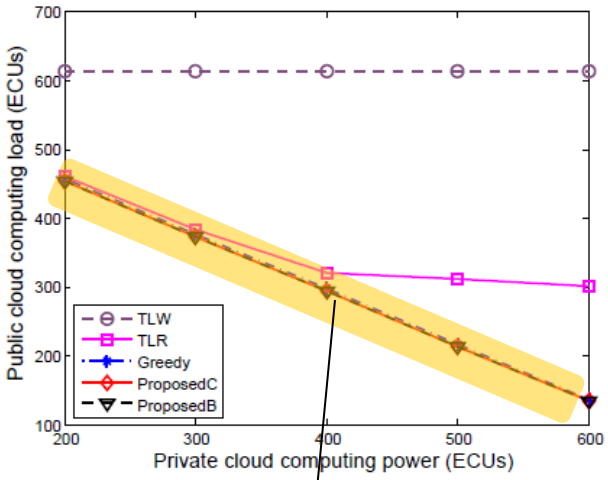
- **Without security constraint**



(a) Monetary cost



(b) Inter-cloud bandwidth



(c) Outsourced computation

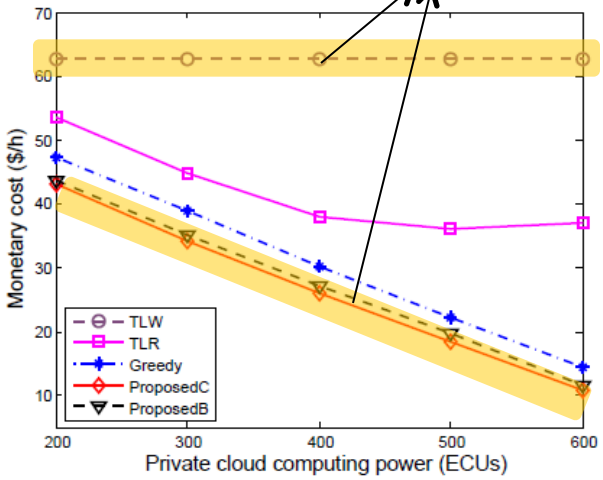
Greedy incurs higher bandwidth cost than ours, and hence higher money cost

Greedy can fully utilizes the private cloud as ours

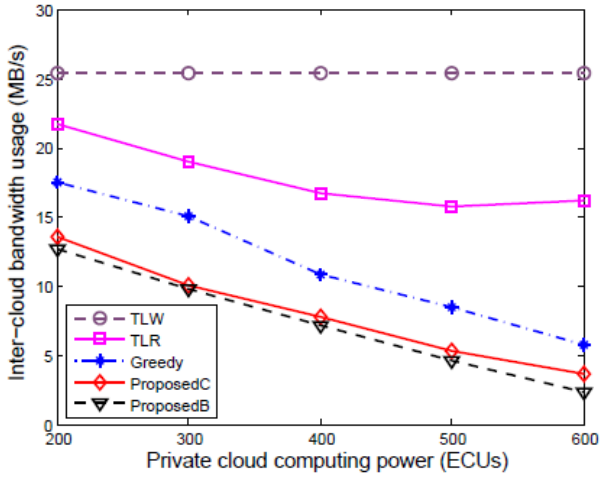
4. Evaluation: simulations

- **Without security constraint**

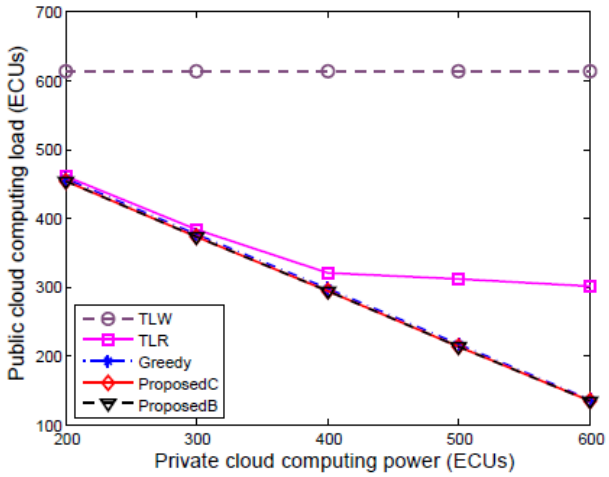
Our schedulers can reduce monetary cost by around 29%-84% compared to a pure public cloud setting



(a) Monetary cost



(b) Inter-cloud bandwidth

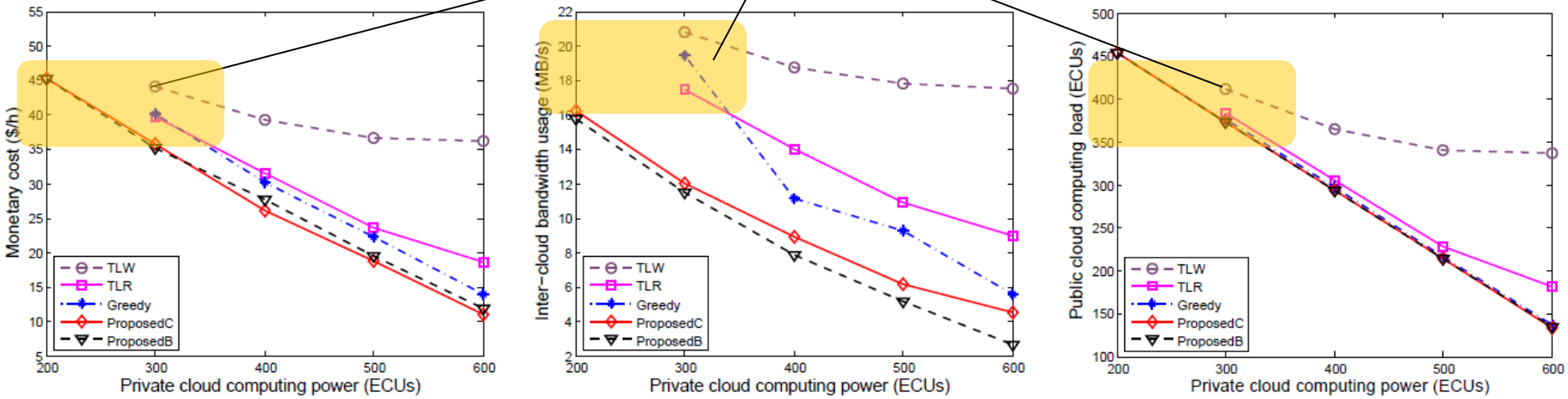


(c) Outsourced computation

4. Evaluation: simulations

- **randomly tags streams to be sensitive**

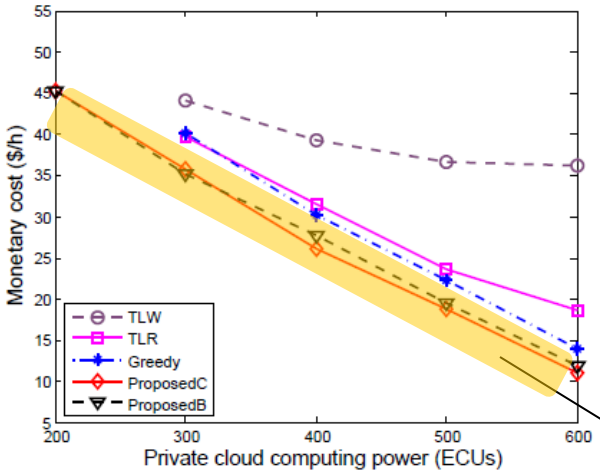
TLW, TLR and Greedy cannot schedule all the tasks when C = 200



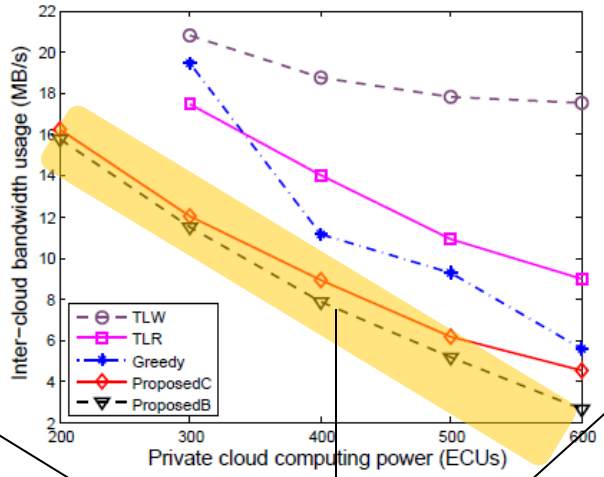
(a) Monetary cost (b) Inter-cloud bandwidth (c) Outsourced computation

4. Evaluation: simulations

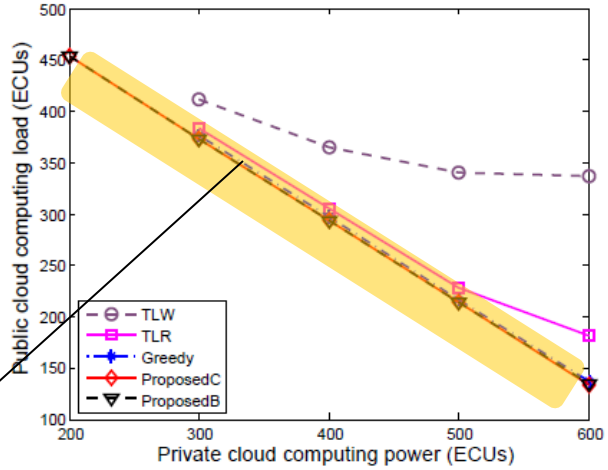
- **randomly tags streams to be sensitive**



(a) Monetary cost



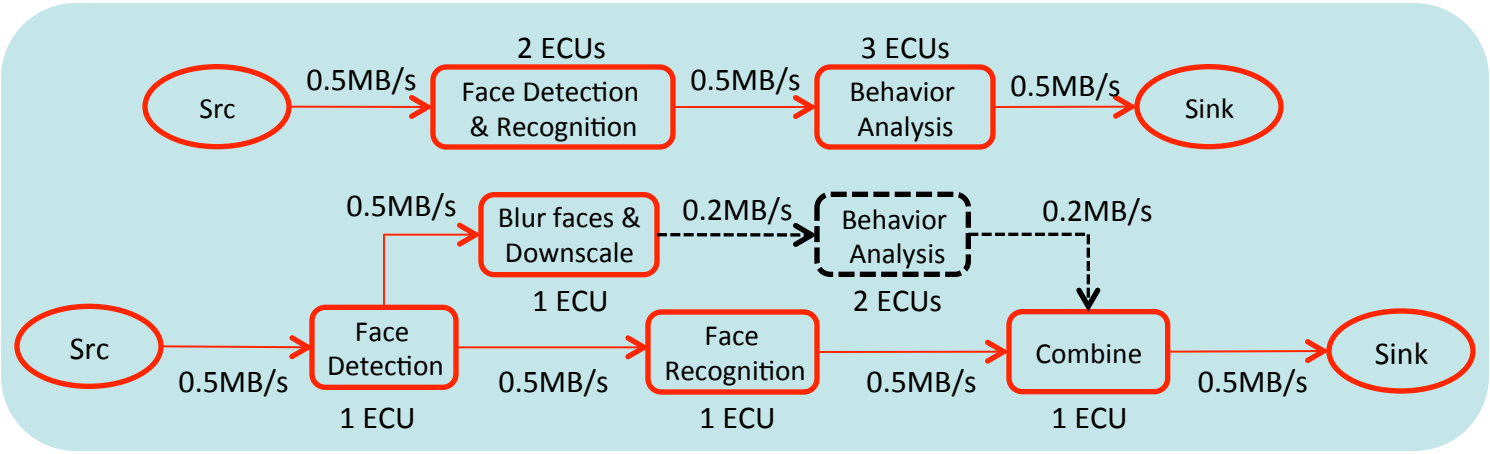
(b) Inter-cloud bandwidth



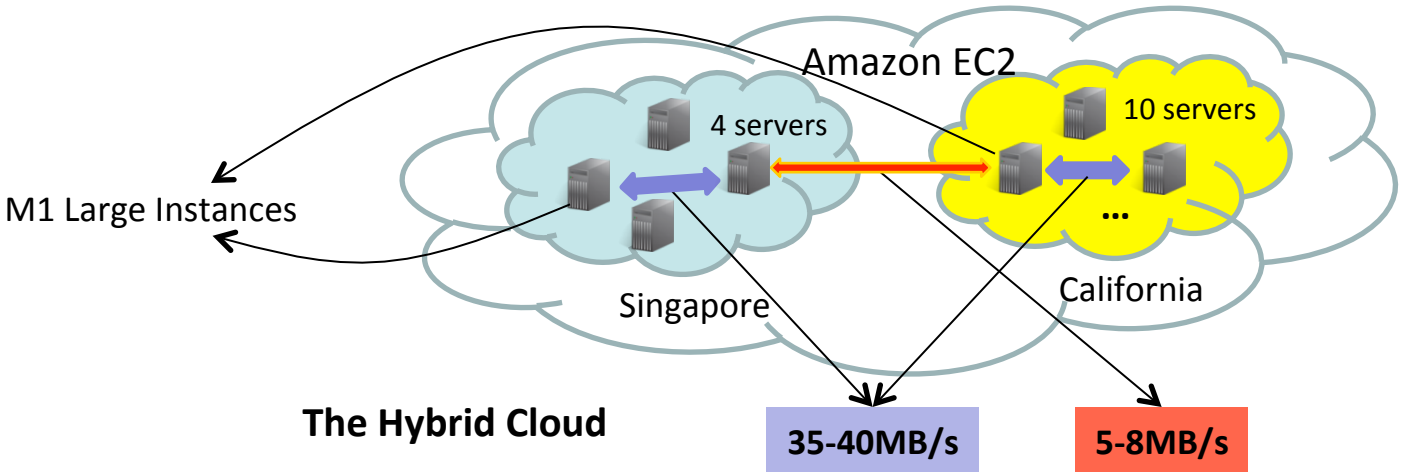
(c) Outsourced computation

Our schedulers can handle all, and constantly outperform the others

4. Evaluation: proof-of-concept system

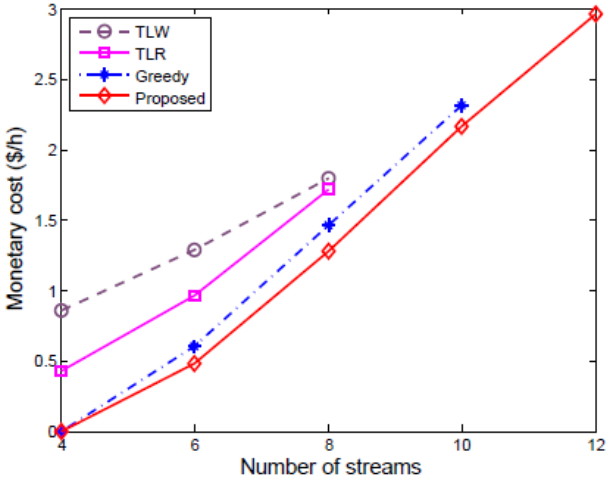


The Task Template (has 2 alternative ways to complete)

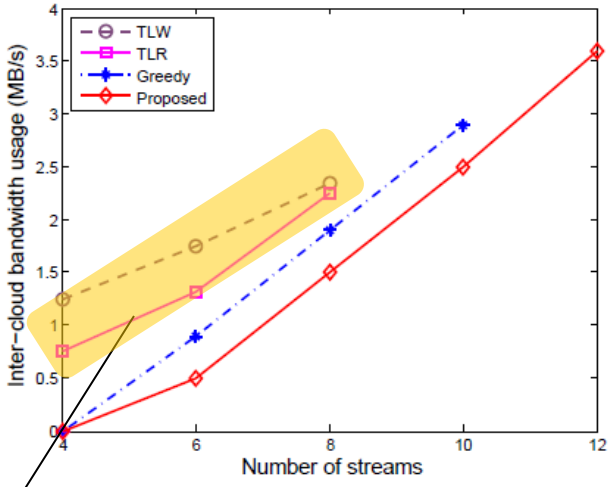


The Hybrid Cloud

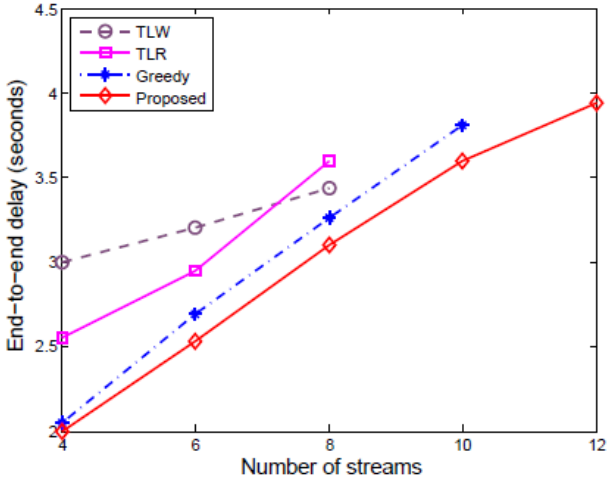
4. Evaluation



(a) Monetary cost



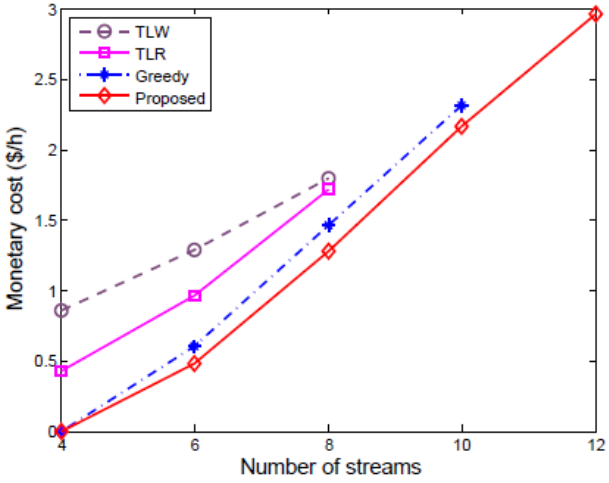
(b) Inter-cloud bandwidth



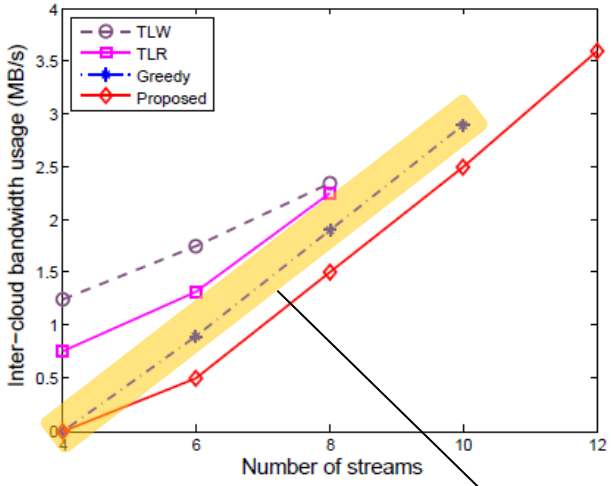
(c) Average end-to-end delay

TLW and TLR fail when
# of streams > 8

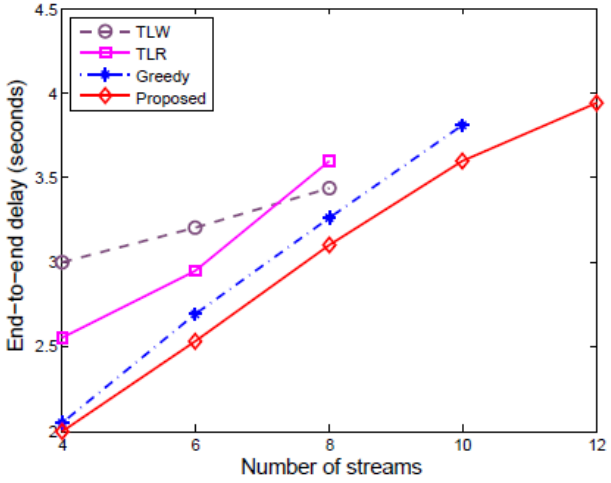
4. Evaluation



(a) Monetary cost



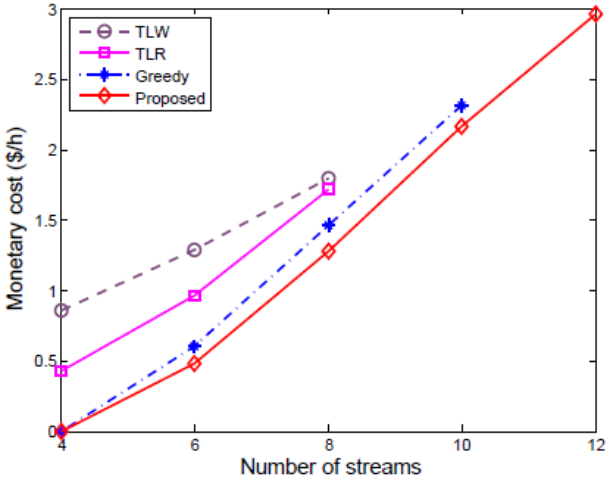
(b) Inter-cloud bandwidth



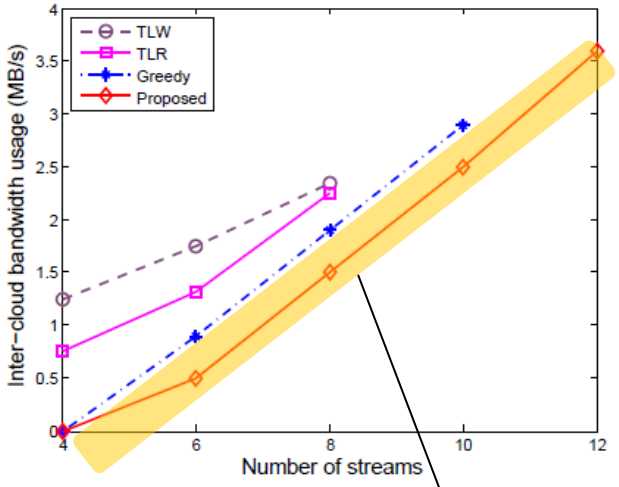
(c) Average end-to-end delay

Greedy also fails when # of streams > 10

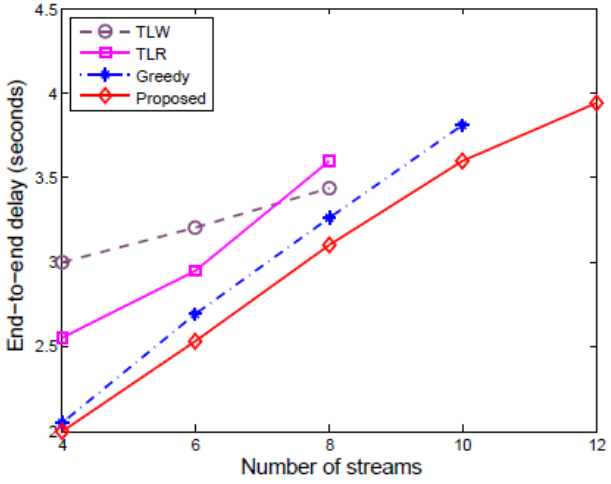
4. Evaluation



(a) Monetary cost



(b) Inter-cloud bandwidth



(c) Average end-to-end delay

ProposedB and ProposedC always choose the same confs, and rendered as one line

5. Conclusions

- **Practical to process large mixed-sensitivity video surveillance streams on hybrid clouds**
- **The proposed scheduler is effective in reducing monetary cost and inter-cloud bandwidth usage**
- **The monetary costs are lower than a single public cloud setting**
- **Future Work**
  - To support real-time re-scheduling
  - To implement on top of existing stream processing systems, e.g., Apache Storm[1]

[1] https://storm.incubator.apache.org/

Q & A