# Improving the scalability of an Application-Level Multicast Protocol

Ayman EL-SAYED

***Vincent ROCA***

[ayman.elsayed|vincent.roca]@inrialpes.fr

Planète project; INRIA Rhône-Alpes

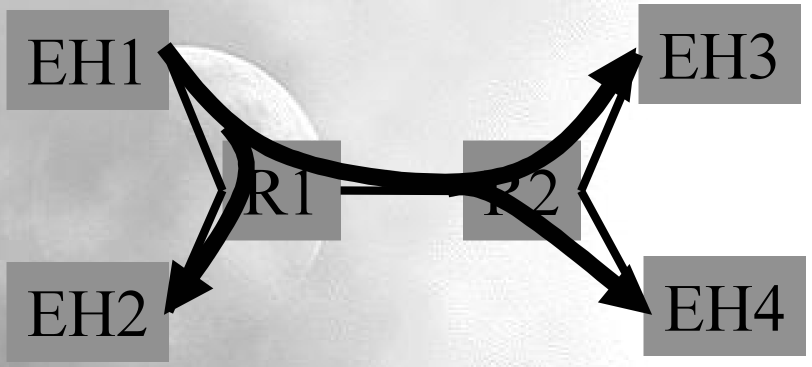
February 25th, 2003

## Introduction to application-level multicast

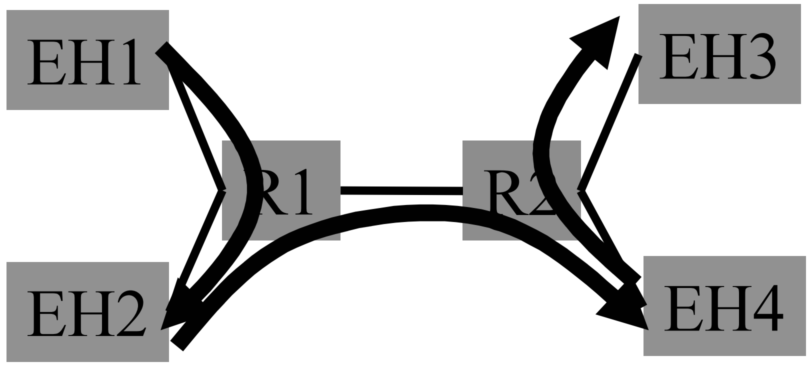
- motivations for application-level multicast
  - multicast routing is not available everywhere
- application-level multicast
  - shifts the multicast support from core routers to end-systems
  - automatic creation an overlay topology
    - **uses unicast between two end-systems**
    - **the underlying physical topology is hidden**
    - **try to find an ``optimal'' overlay topology (e.g. a spanning tree with minimal global cost)**

## Introduction to ALM… (cont')

- example



*with multicast routing*



*with application-level multicast*

- topology building algorithm can be:
  - *centralized (HBM, ALMI …)*
  - distributed ( NARADA, Overcast, Nice, TBCP...)

## Introduction to ALM… (cont')

- requires a dynamic overlay topology update
  - because the networking conditions dynamically change
    - **try to stay as close as possible to an optimal overlay topology**
    - **can be regarded as ``static QoS routing''**
  - because the group is dynamic, the topology becomes sub-optimal
    - **after a node departure/failure, a quick and dirty local solution is found to avoid topology partitioning**
    - **when a node arrives, he joins the current topology as a leaf to create as little perturbation as possible**

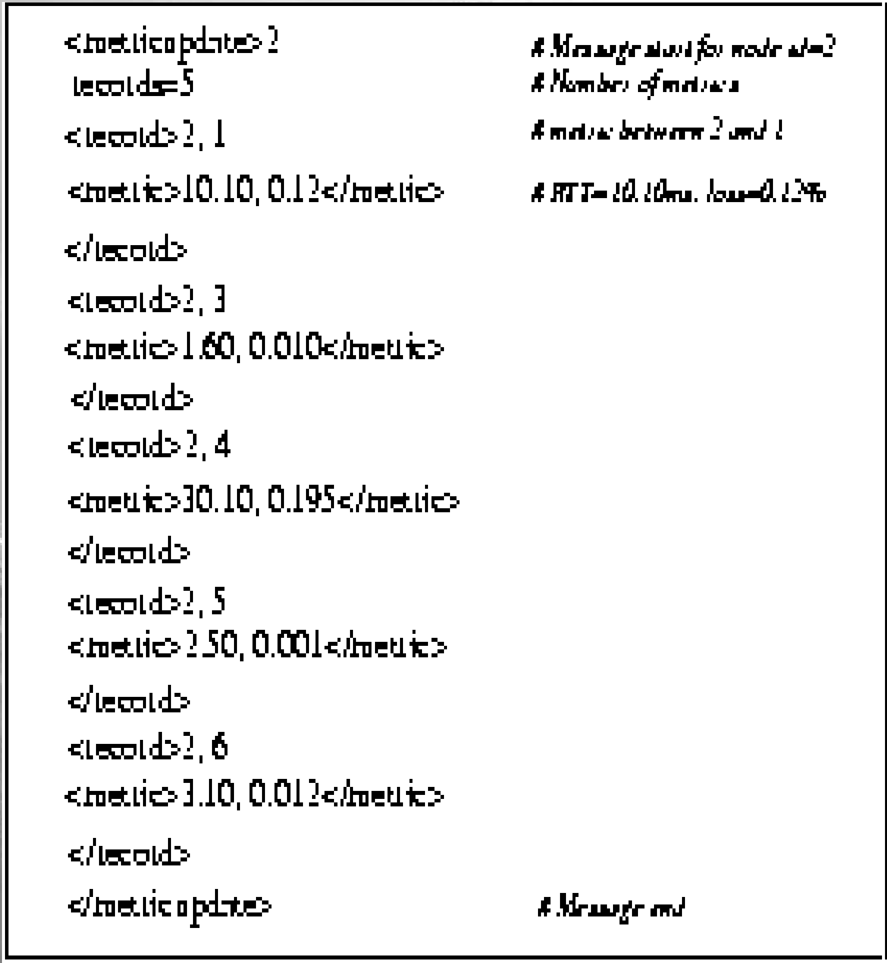
- we need to periodically update the whole topology !

## Our HBM application-level multicast

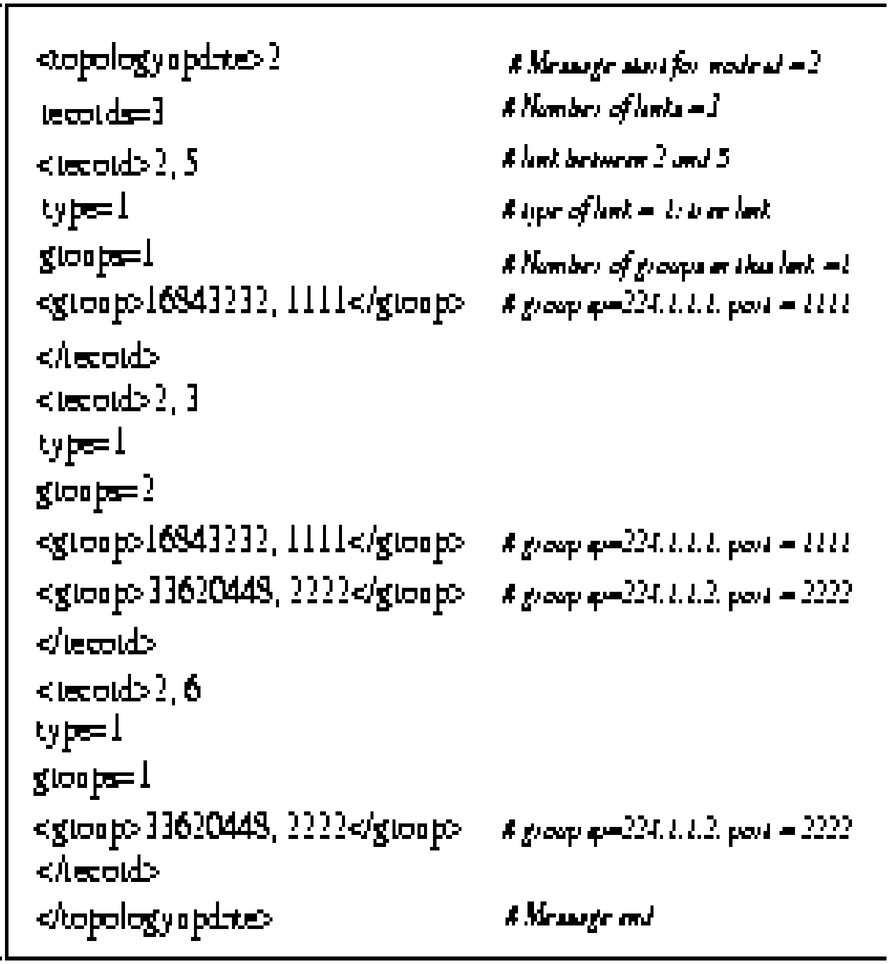
- centralized approach:
    - everything is controlled by a Rendez-vous Point (RP)
    - the RP has a complete knowledge of group membership/communication costs
    - take into account several metrics (RTT, loss) when creating the virtual topology
    - data flows on the virtual topo. (no RP implication)
- each node periodically evaluates metrics between itself and other nodes and informs the RP ( period: $T_{mu}$).
- the RP periodically refreshes the topology and informs all nodes (period: $T_{tu}$).

# Our HBM proposal… (cont’)

metric update msg (MU) topo. update msg (TU)



```
<metricupdate>2                         # Message start for node id=2
 records=5                              # Number of metrics
<record>2, 1                            # metric between 2 and 1
<metric>10.10, 0.12</metric>            # RTT=10.10ms, loss=0.12%
</record>
<record>2, 3
<metric>1.60, 0.010</metric>
 </record>
<record>2, 4
<metric>30.10, 0.195</metric>
</record>
<record>2, 5
<metric>2.50, 0.001</metric>
</record>
<record>2, 6
<metric>3.10, 0.012</metric>
</record>
</metricupdate>                         # Message end
```



```
<topologyupdate>2                        # Message start for node id = 2
 records=3                               # Number of links = 3
<record>2, 5                             # link between 2 and 5
 type=1                                  # type of link = 1: is an link
groups=1                                 # Number of groups on this link = 1
<group>16843232, 1111</group>            # group ip=224.1.1.1, port = 1111
</record>
<record>2, 3
type=1
groups=2
<group>16843232, 1111</group>            # group ip=224.1.1.1, port = 1111
<group>33620448, 2222</group>            # group ip=224.1.1.2, port = 2222
</record>
<record>2, 6
type=1
groups=1
<group>33620448, 2222</group>            # group ip=224.1.1.2, port = 2222
</record>
</topologyupdate>                        # Message end
```

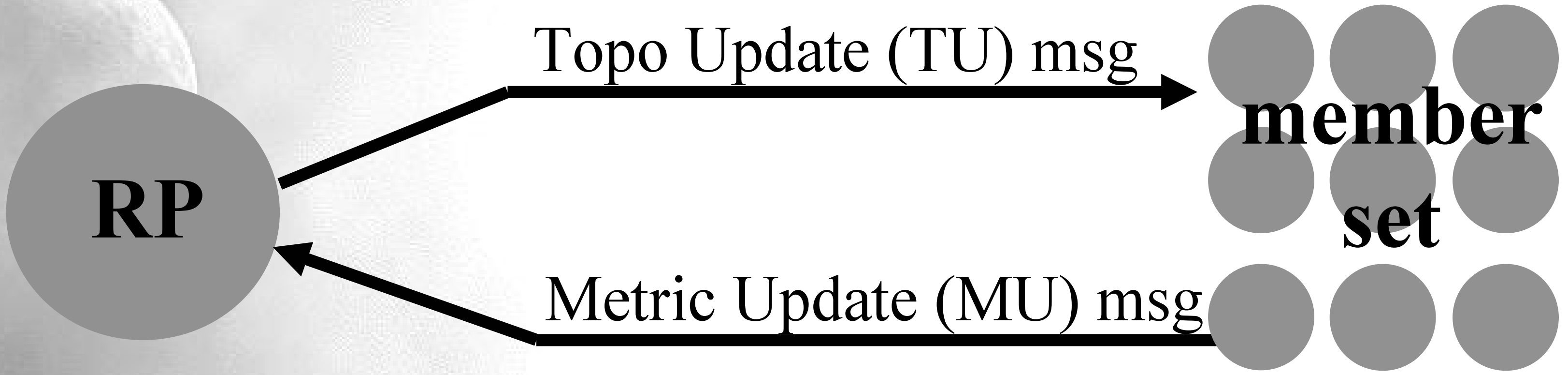
both follow an XML format (simplicity, flexibility)

## Our HBM proposal… (cont')

- centralized      scalability problems limit the topology update frequency
- goal:

  how can we improve this scalability ?
- solution: reduce the control traffic overhead

## Improving the scalability



- Total control traffic rate = $R_{cntl}(N) = R_{tu}(N) + R_{mu}(N)$
- MU control rate = $R_{mu}(N)$
- TU control rate = $R_{tu}(N)$
- Number of Members = N

## Improving the scalability… (cont')

- message sizes
  - size of MU msg = $S_{mu}(N)$
  - number of MU records per msg = $n_{rmu}(N)$
  - mean size of a MU record = $S_{rmu}$
  - size of a TU msg = $S_{all\_tu}(N)$

$$S_{mu}(N) = S_{mu\_header} + n_{rmu}(N) * S_{rmu}$$

$$R_{mu}(N) = \frac{N * s_{mu}(N)}{T_{mu}(N)}$$

$$Sla\ \_\ tu = N * Stu\ \_\ hd + 2 * h * Srtu$$

$$R_{tu}(N) = \frac{s_{all\_tu}(N)}{T_{tu}(N)}$$

$$R_{ctrl}(N) = R_{mu}(N) + R_{tu}(N)$$

## Improving the scalability… (cont')

- principle
  - keep the total control traffic rate inferior to a given percentage of the total session traffic
    - **(same strategy as RTCP)**

$$R_{ctrl}(N) \leq \alpha * (R_{ctrl}(N) + R_{data})$$

    - **in our experiments: $\alpha$ = 5%, $R_{data}$= 128 kbps**

- play with the various protocol parameters

$$\left.\begin{array}{ll}\textbf{metric update refresh period:} & T_{mu} \\ \textbf{number of records per MU message} & \end{array}\right\} \text{EH}$$

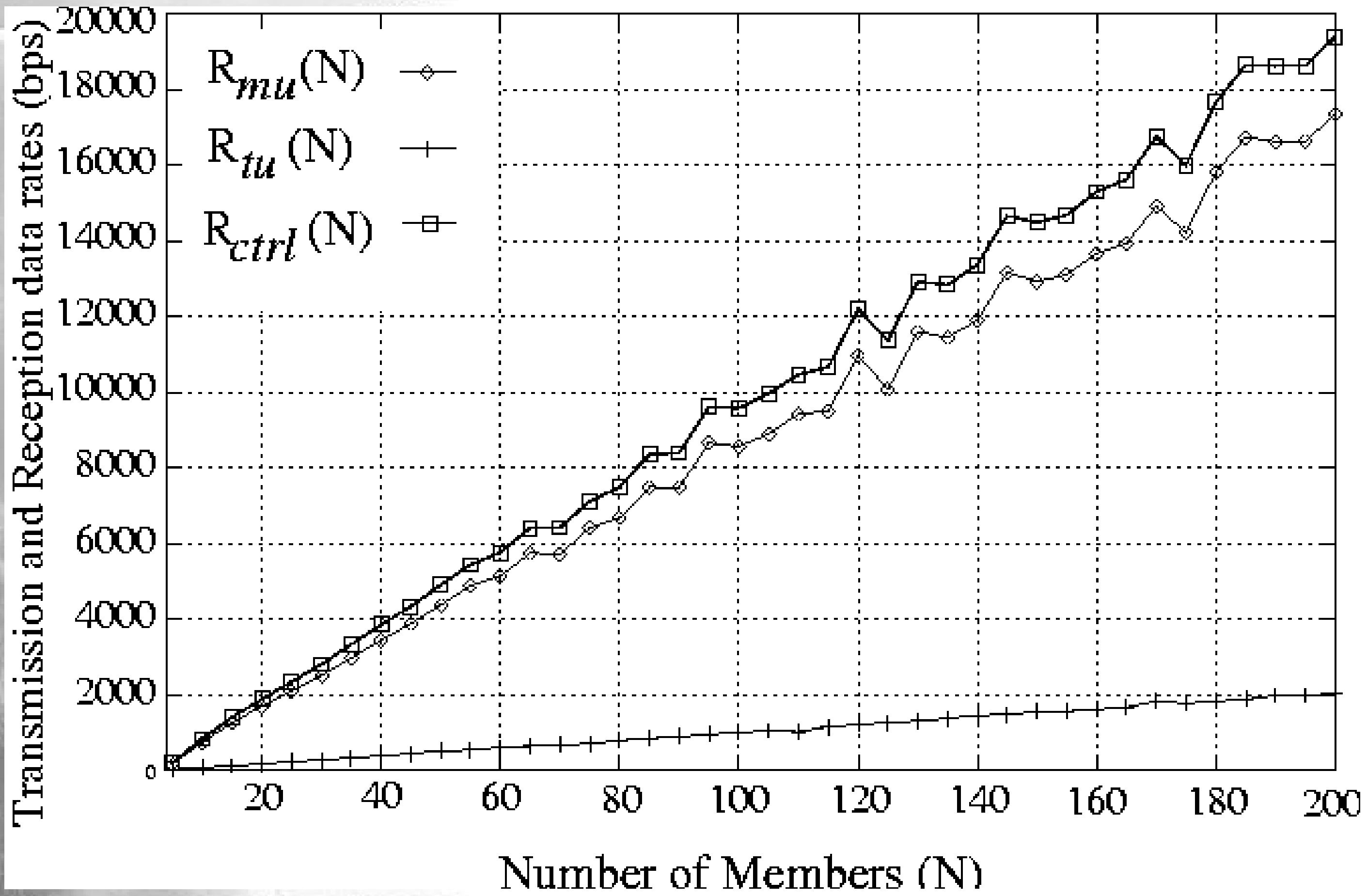
$$\left.\begin{array}{ll}\textbf{topology refresh period:} & T_{tu}\end{array}\right\} \text{RP}$$

## Improving the scalability… (cont’)

- we defined 4 strategies
  - strategy 1 is the reference
    - **does not include any optimization**
  - strategies 2, 3 and 4 differ on the way the $T_{mu}$, $N_{rmu}$, $T_{tu}$ parameters are managed
  - they all achieve their goals !

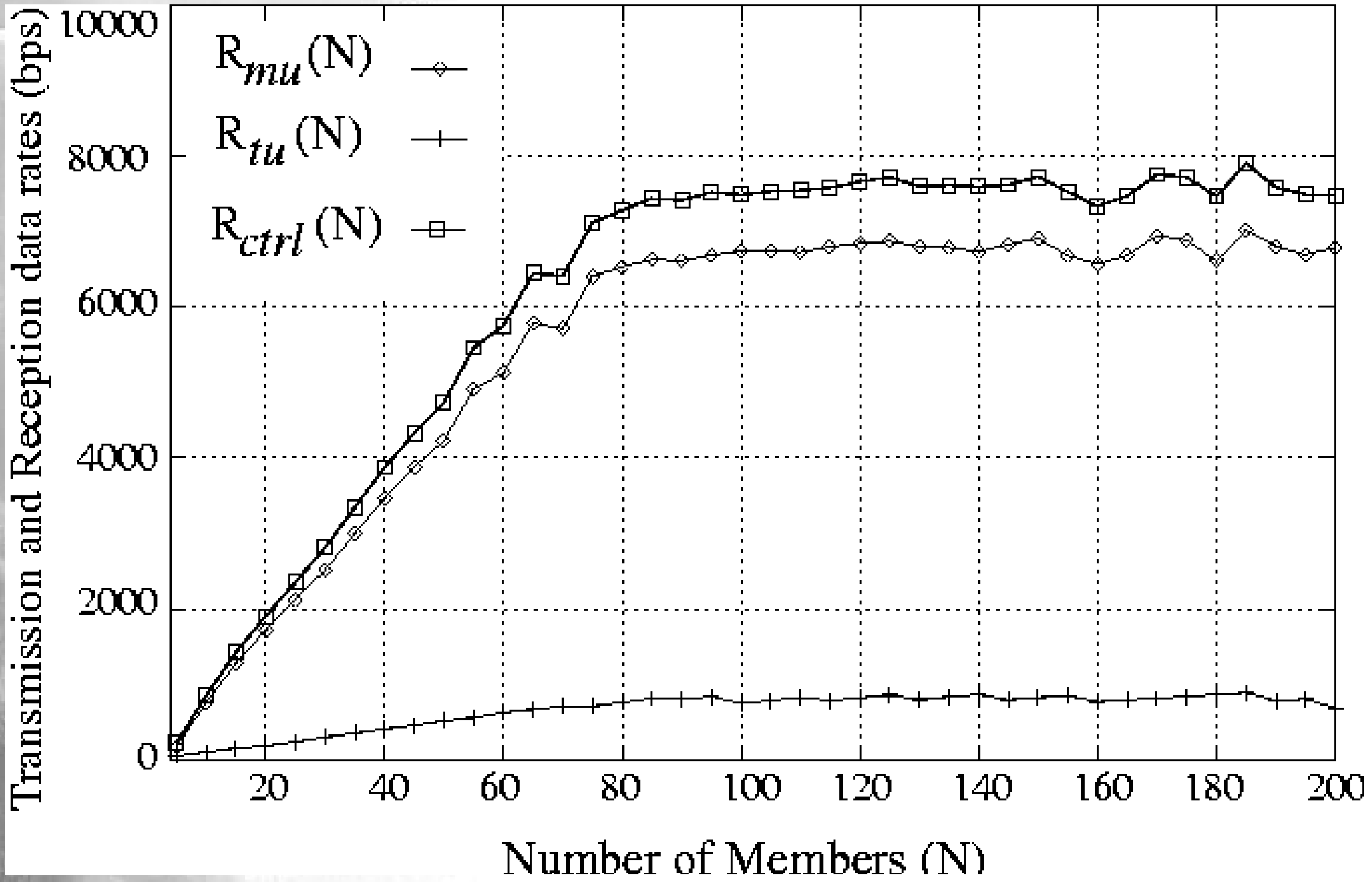
## Strategy 1: reference

- no control traffic bandwidth limitation!



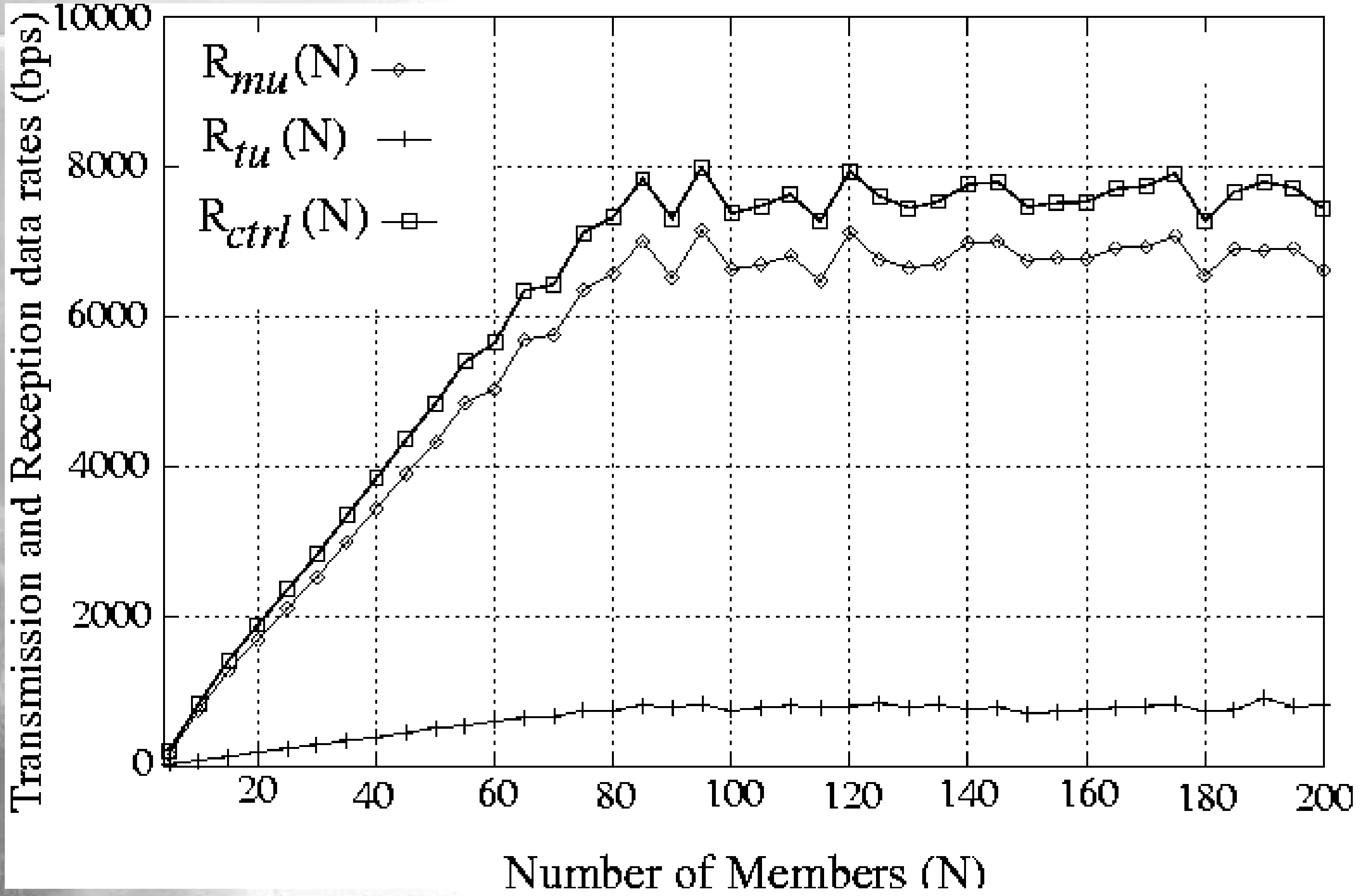
## Strategy 2

- achieves control traffic bandwidth limitation!



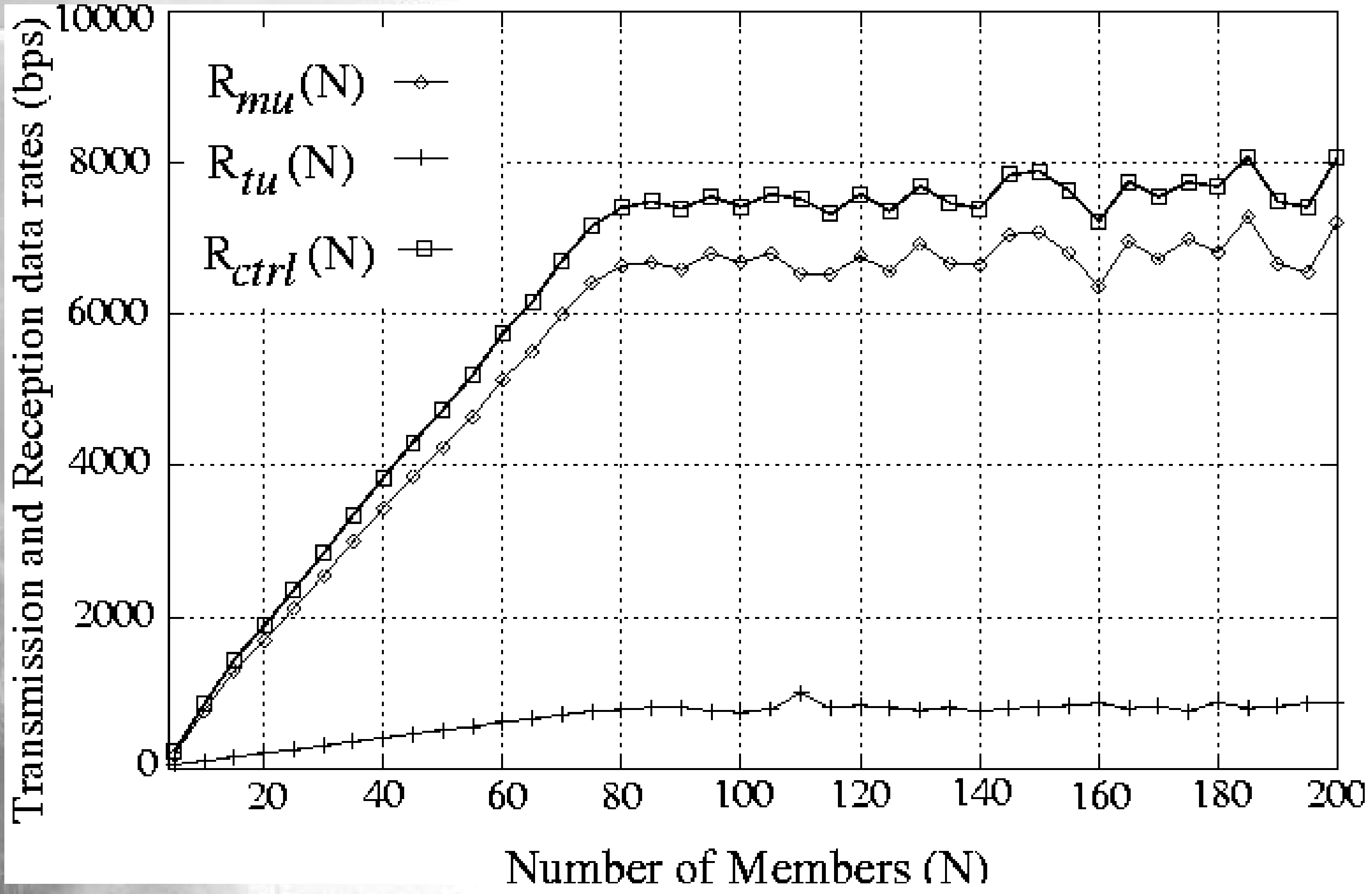
## Strategy 3

- achieves control traffic bandwidth limitation!



## Strategy 4

- achieves control traffic bandwidth limitation!



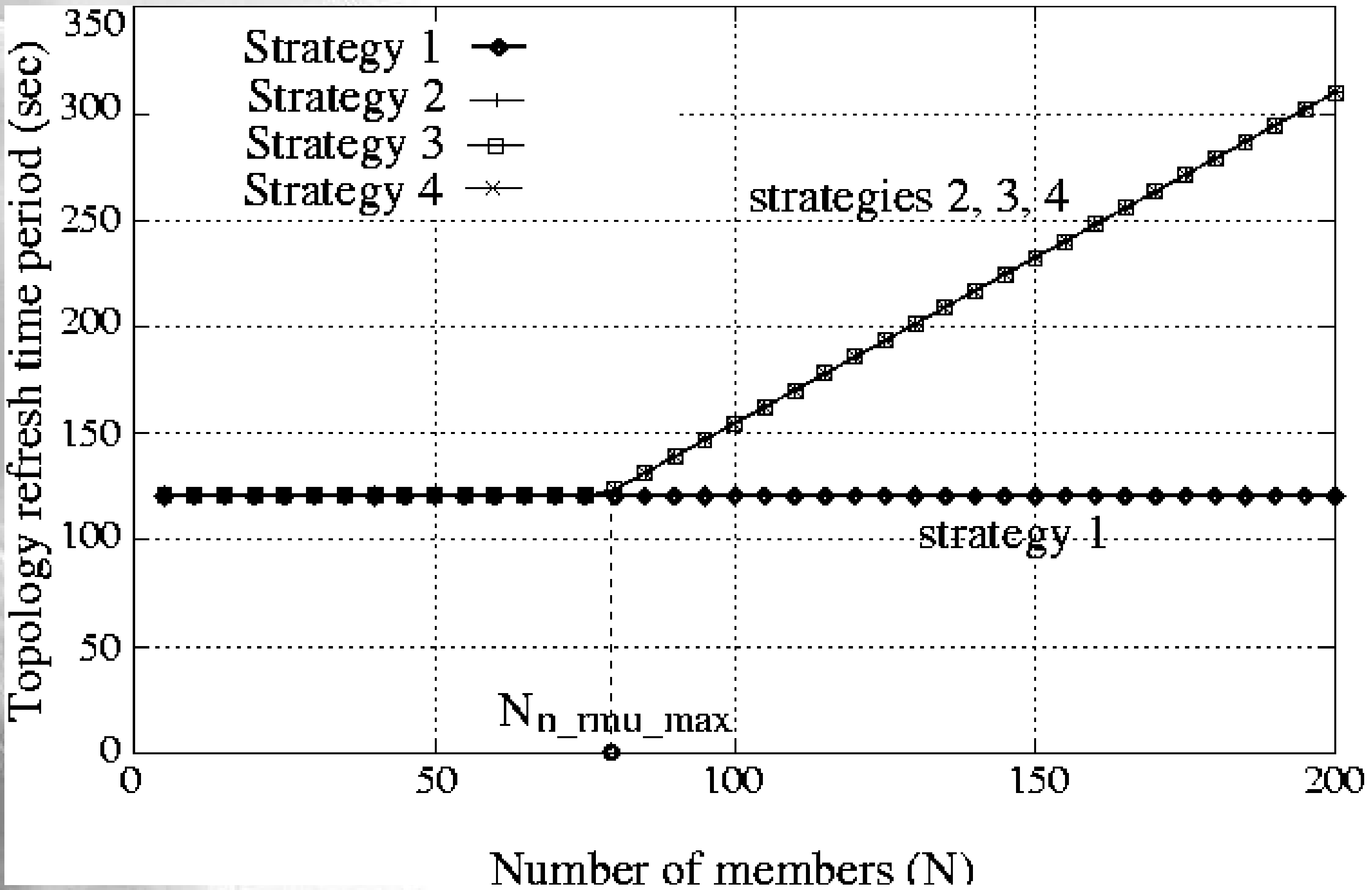
## Improving the scalability… (cont')

- since all strategies achieve their goal, the question is now:

  "what is the best way to achieve this goal?''

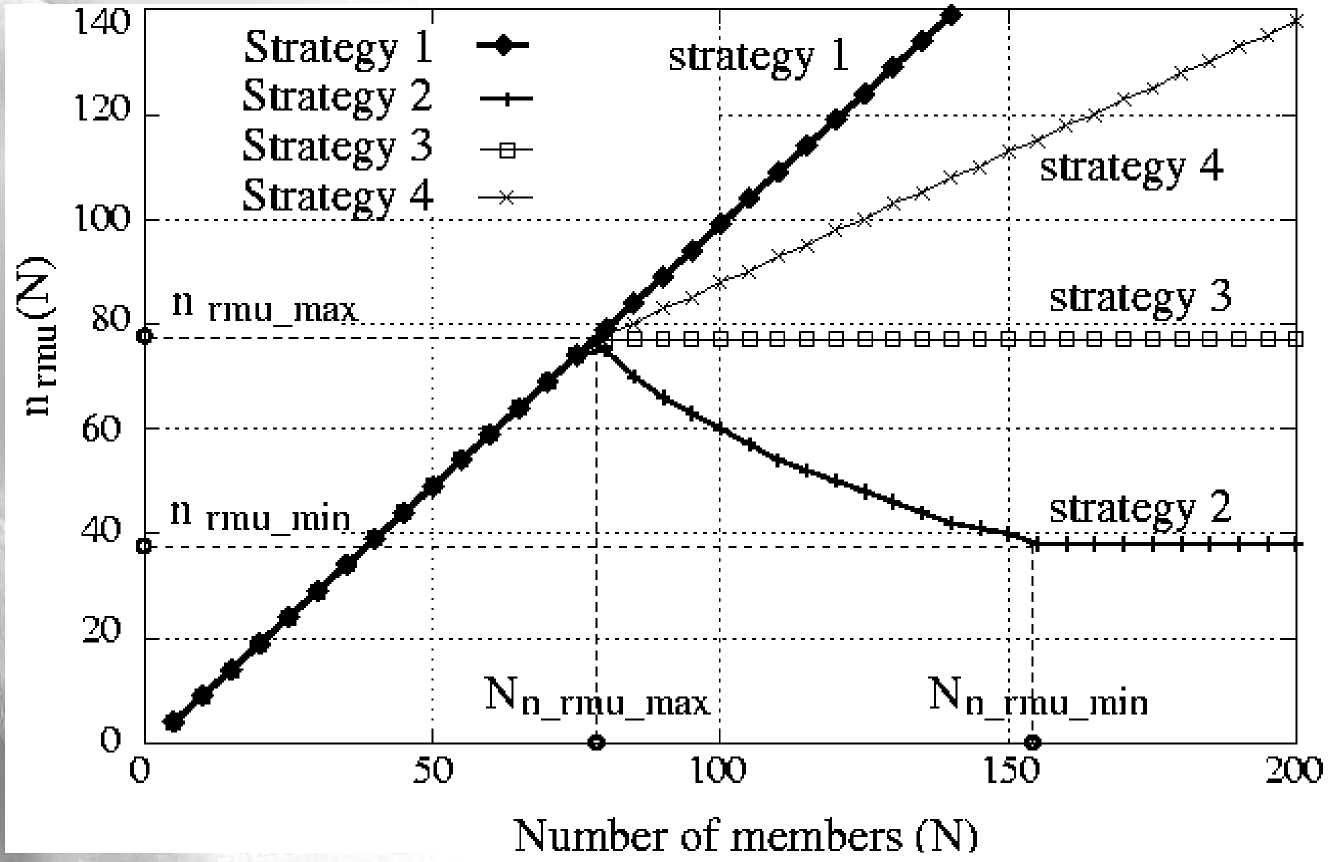
## $T_{tu}(N)$: outgoing control rate

- no difference between strategies 2, 3 and 4



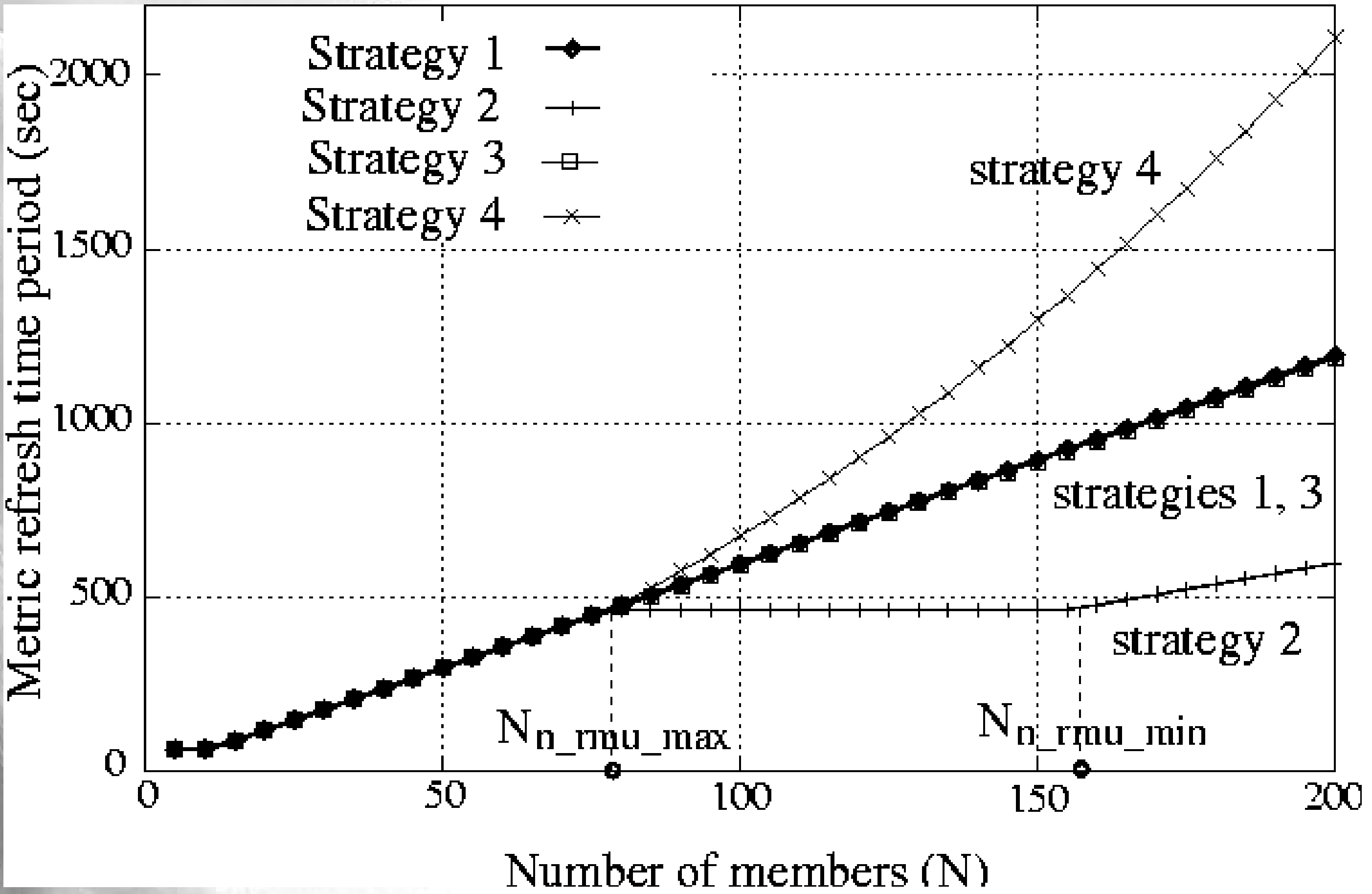
## $N_{rmu}$( N): nb of records in a MU message

- $N_{rmu}$(N) must be limited above a given threshold



## $T_{mu}(N)$: MU refresh period

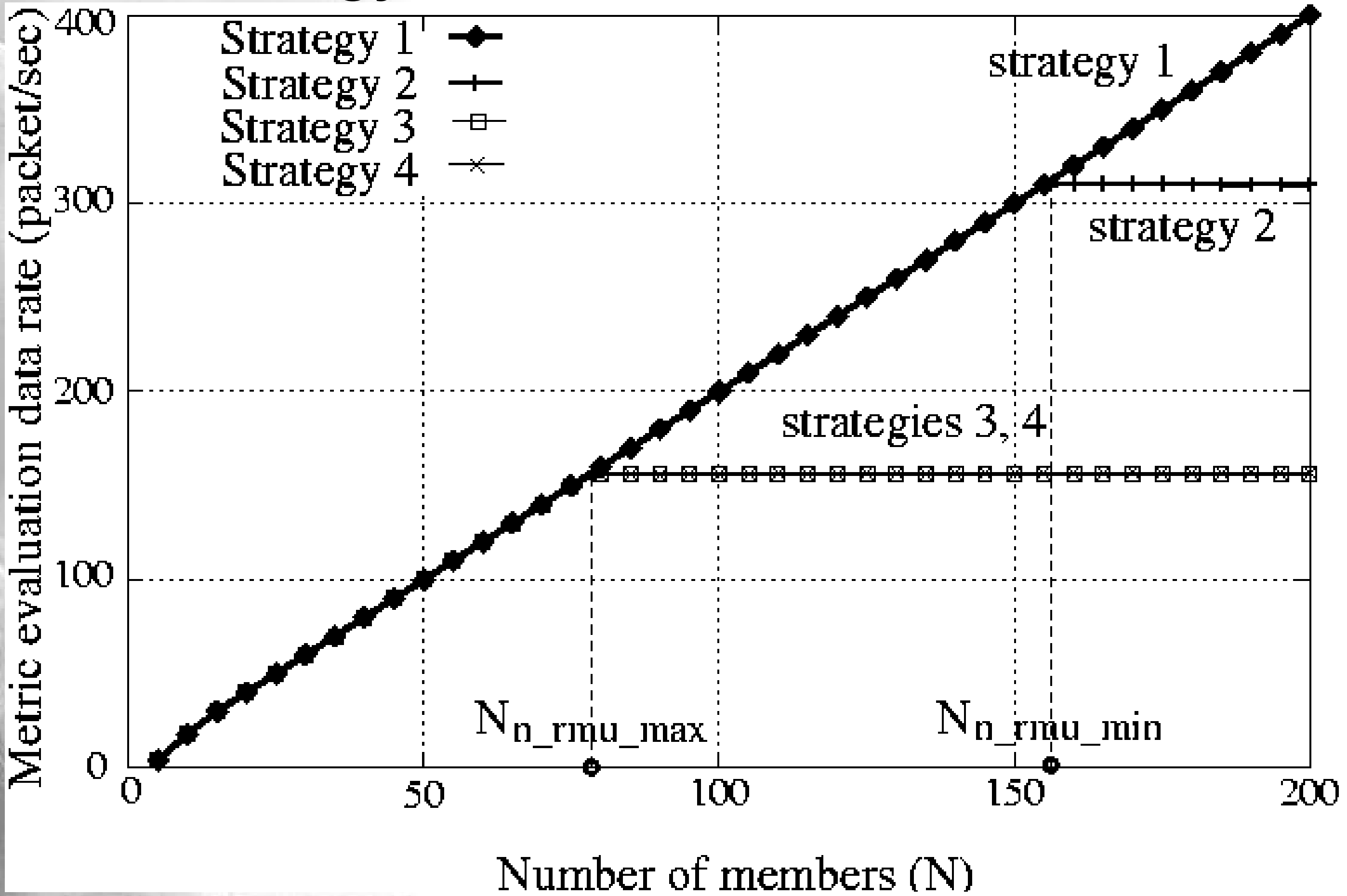
- strategy 4 is bad because of high $T_{mu}(N)$



- now comparison is between strategies 2 & 3

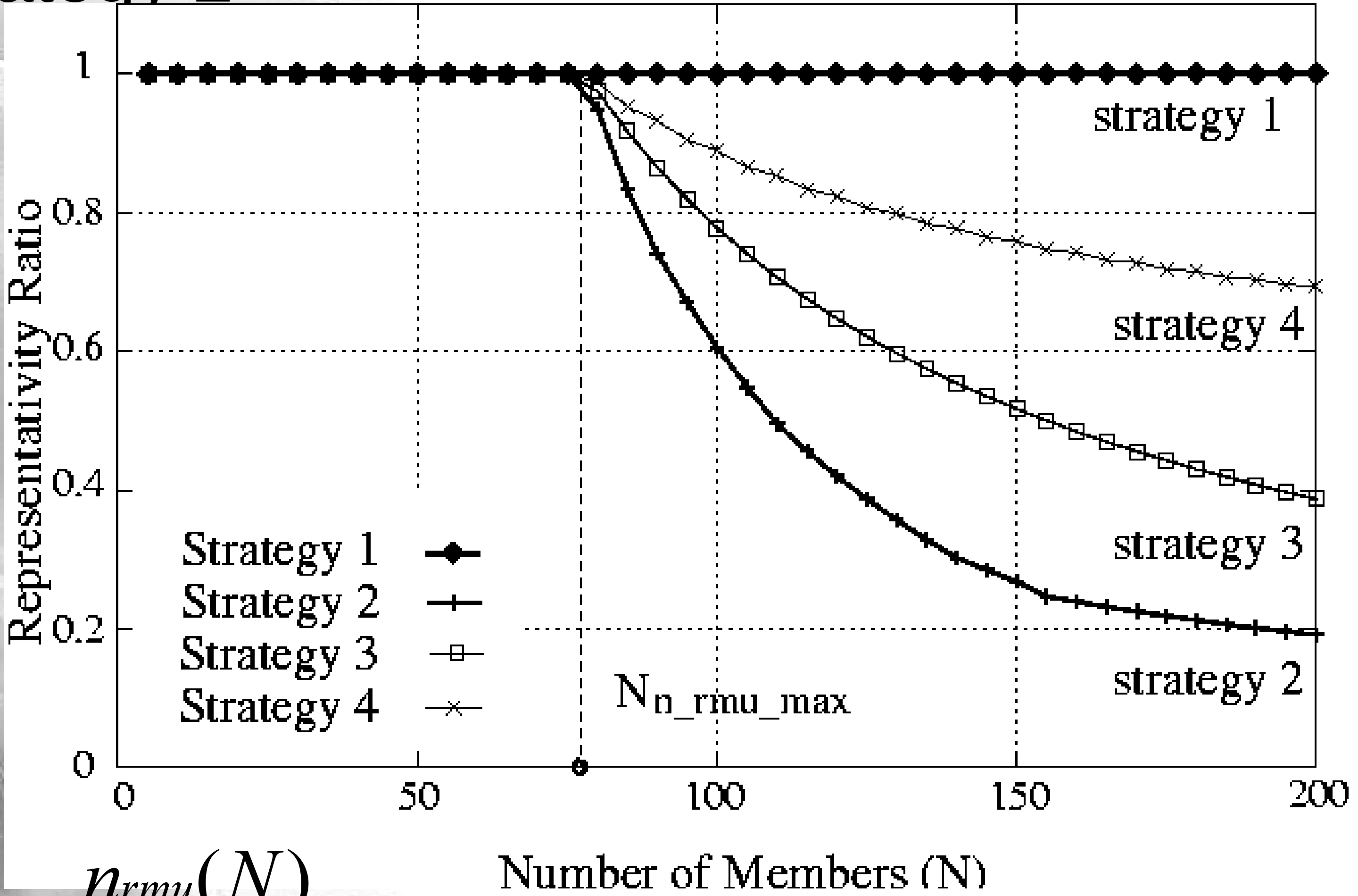
## Metric evaluation overhead

- metric evaluation of strategy 3 is better than that of strategy 2



## Representativity ratio

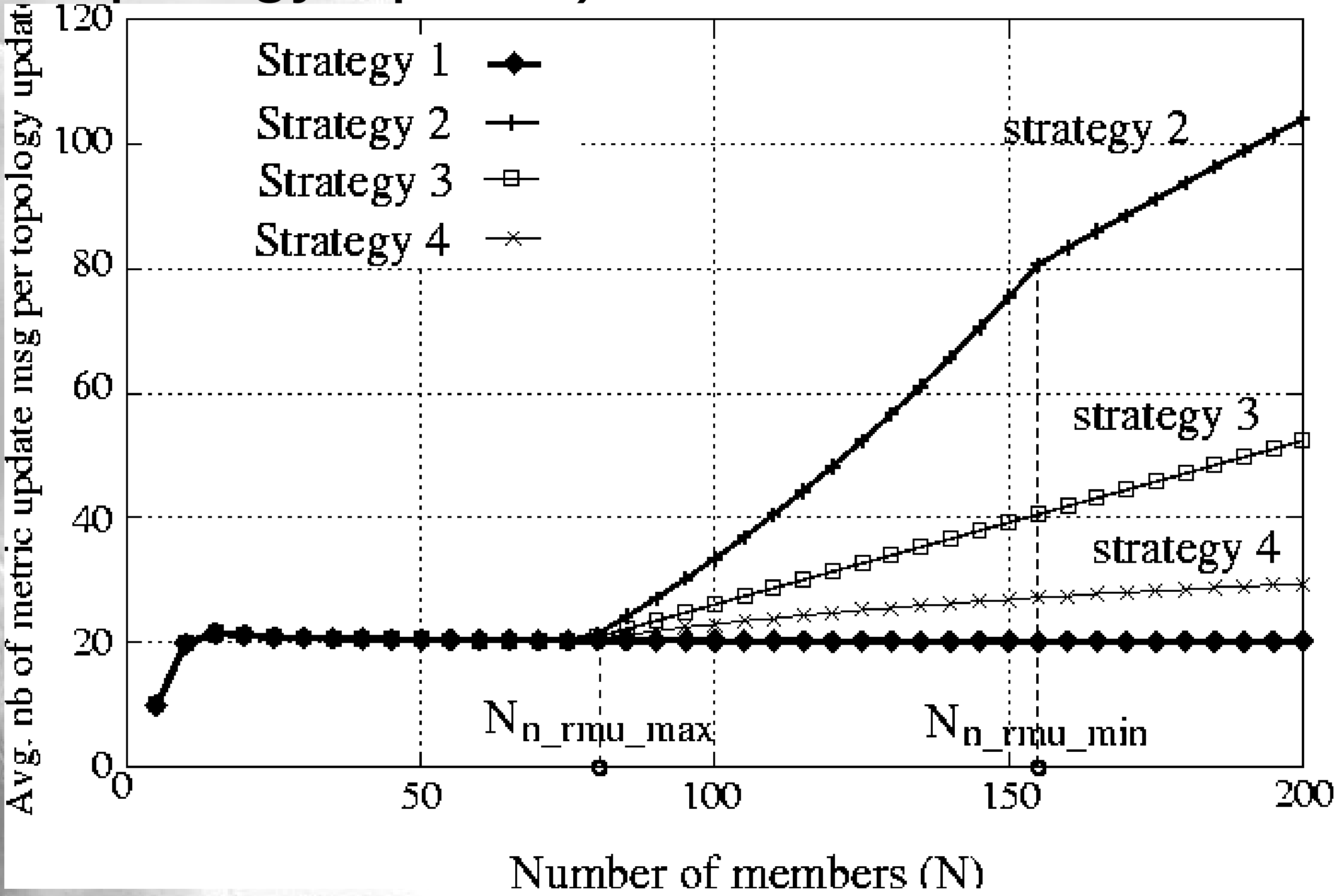
- strategy 3 has a better representativity than strategy 2



$$R = \frac{n_{rmu}(N)}{N-1}$$

## Number of metrics refreshed per TU

- strategy 2 is the best (more metric records per topology update)



## Conclusions

- a centralized application layer multicast
  - everything is controlled by a RP
  - simple and efficient
- to improve the scalability
  - step 1: model of the protocol behavior
  - step 2: define a target (max overhead)
  - step 3: identify the best way to act on the model parameters (strategy 3)
- complementary solutions
  - textual control message compression
  - the number of effective users is larger since a EH can serve many local clients using local native multicast

## The strategies

***** Limit the $n_{rmu}(N)$ *****

- Strategy 2:
  - $n_{rmu}(N)$ decreases, $T_{mu}(N)$ remains constant
  - after a threshold, $n_{rmu}(N)$ remains constant and $T_{mu}(N)$ increases
- Strategy 3:
  - $n_{rmu}(N)$ remains constant, $T_{mu}(N)$ increases
- Strategy 4:
  - $n_{rmu}(N)$ and $T_{mu}(N)$ both increase