A Tighter Recursive Calculus to Compute the Worst-Case Traversal Time of Real-Time Traffic over NoCs

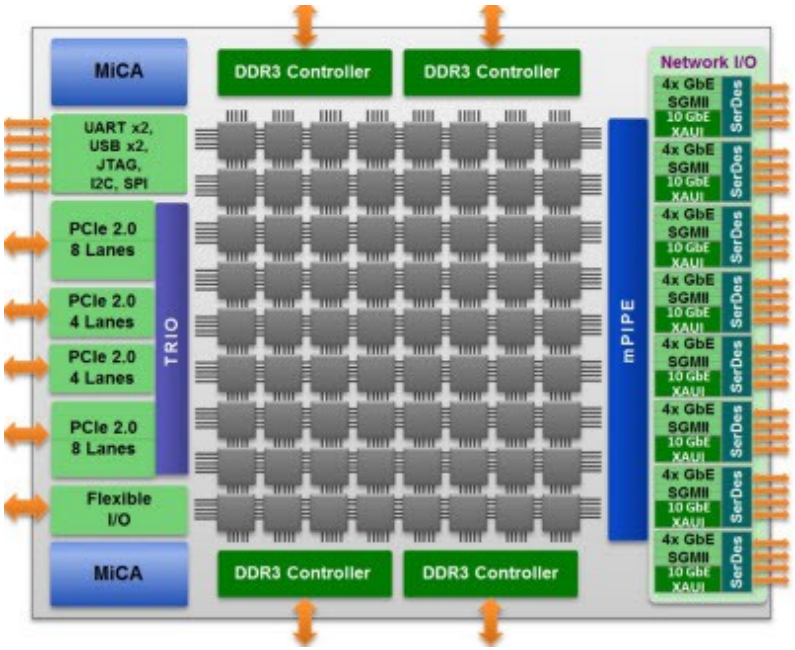
Meng Liu, Matthias Becker, Moris Behnam, Thomas Nolte

ASP-DAC, Chiba, Japan
17. January 2017

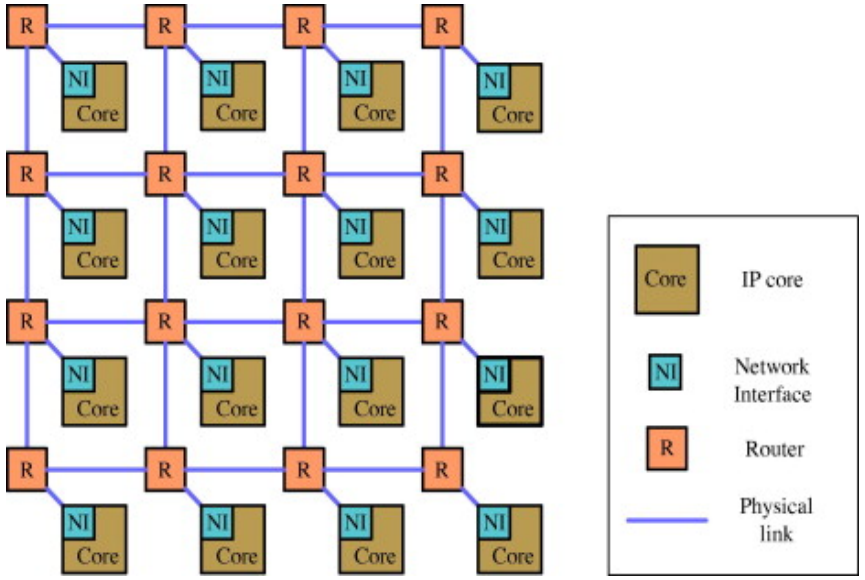
Outline

- Introduction
- Motivation
- Recap - Recursive Calculus
- Modified Recursive Calculus
- Evaluation
- Conclusion

Many-core Platforms and NoCs



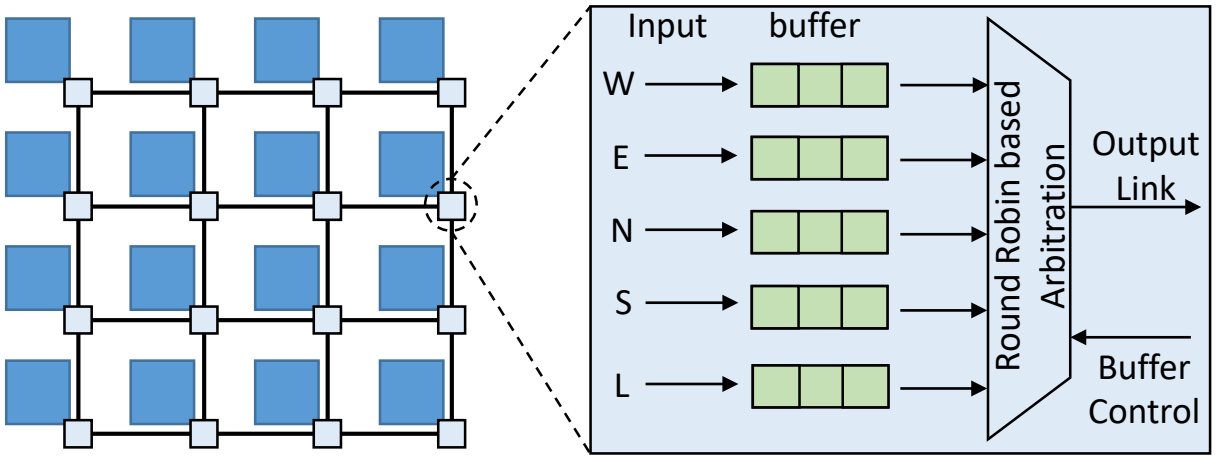
Pics from Tilera



Pic by Nuno Roma

System Model - NoC

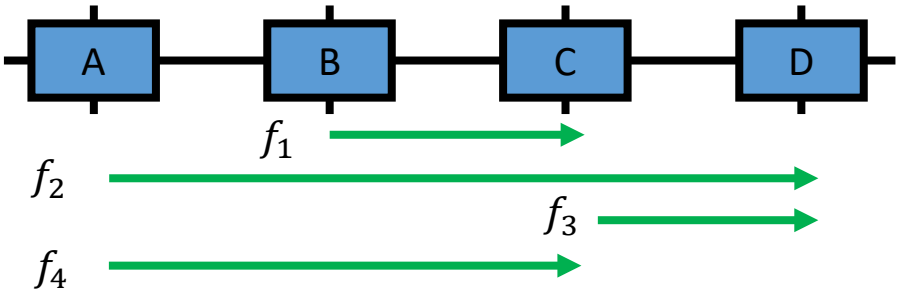
- 2D-mesh based NoC
  - Wormhole-switching
  - Round-robin based
  - XY-routing



System Model - Flow

- Real-time Flows
  - Periodic or sporadic
  - L_i - Packet size
  - T_i - Minimum Inter-arrival Time (MIT)
  - D_i - relative deadline ($D_i \leq T_i$)
  - R_i - fixed route/path

- W_i - Worst-Case Traversal Time of f_i



Motivation (1/3)

- Timeliness is important for real-time applications
  - Each packet should be delivered within its deadline (i.e. $W_i \leq D_i$)

Motivation (1/3)

- Timeliness is important for real-time applications
  - Each packet should be delivered within its deadline (i.e. $W_i \leq D_i$)

- How can a designer verify such a timing requirement?

Motivation (1/3)

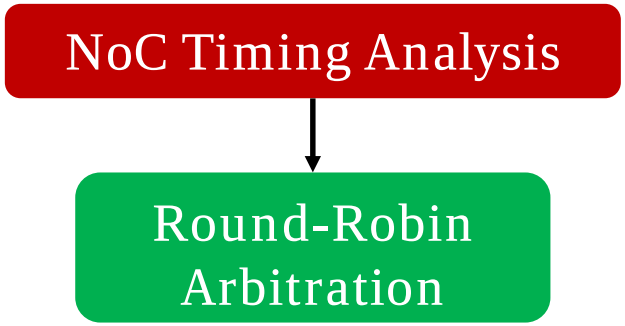
- Timeliness is important for real-time applications
  - Each packet should be delivered within its deadline (i.e. $W_i \leq D_i$)

- How can a designer verify such a timing requirement?
  - Timing Analysis

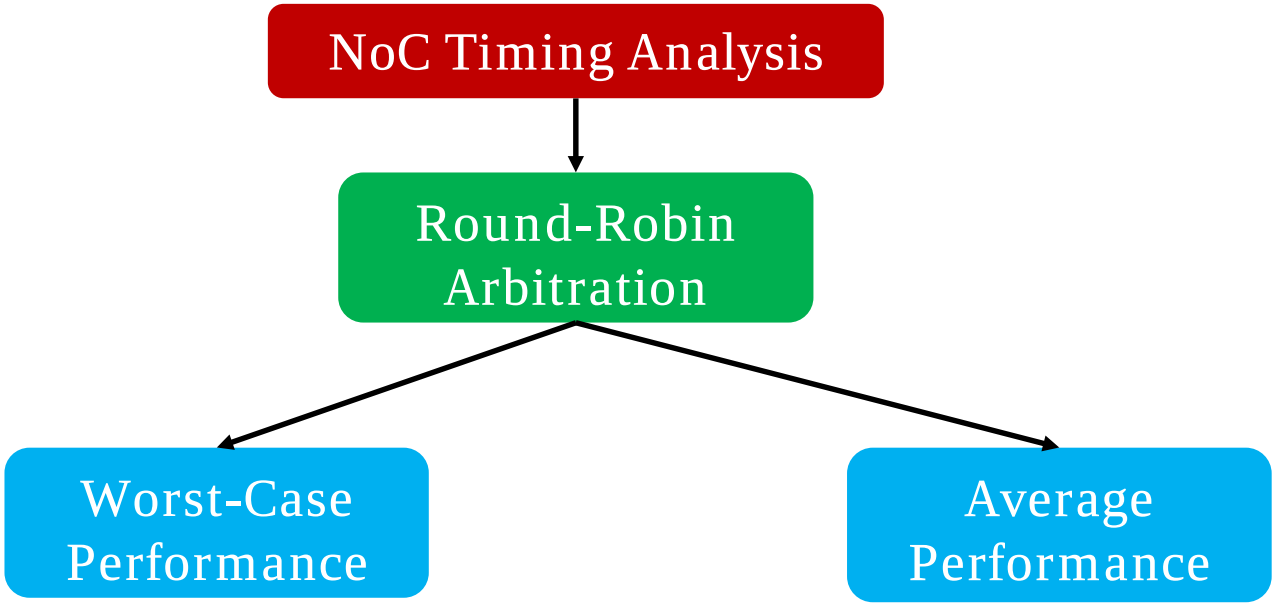
Motivation (2/3)

NoC Timing Analysis

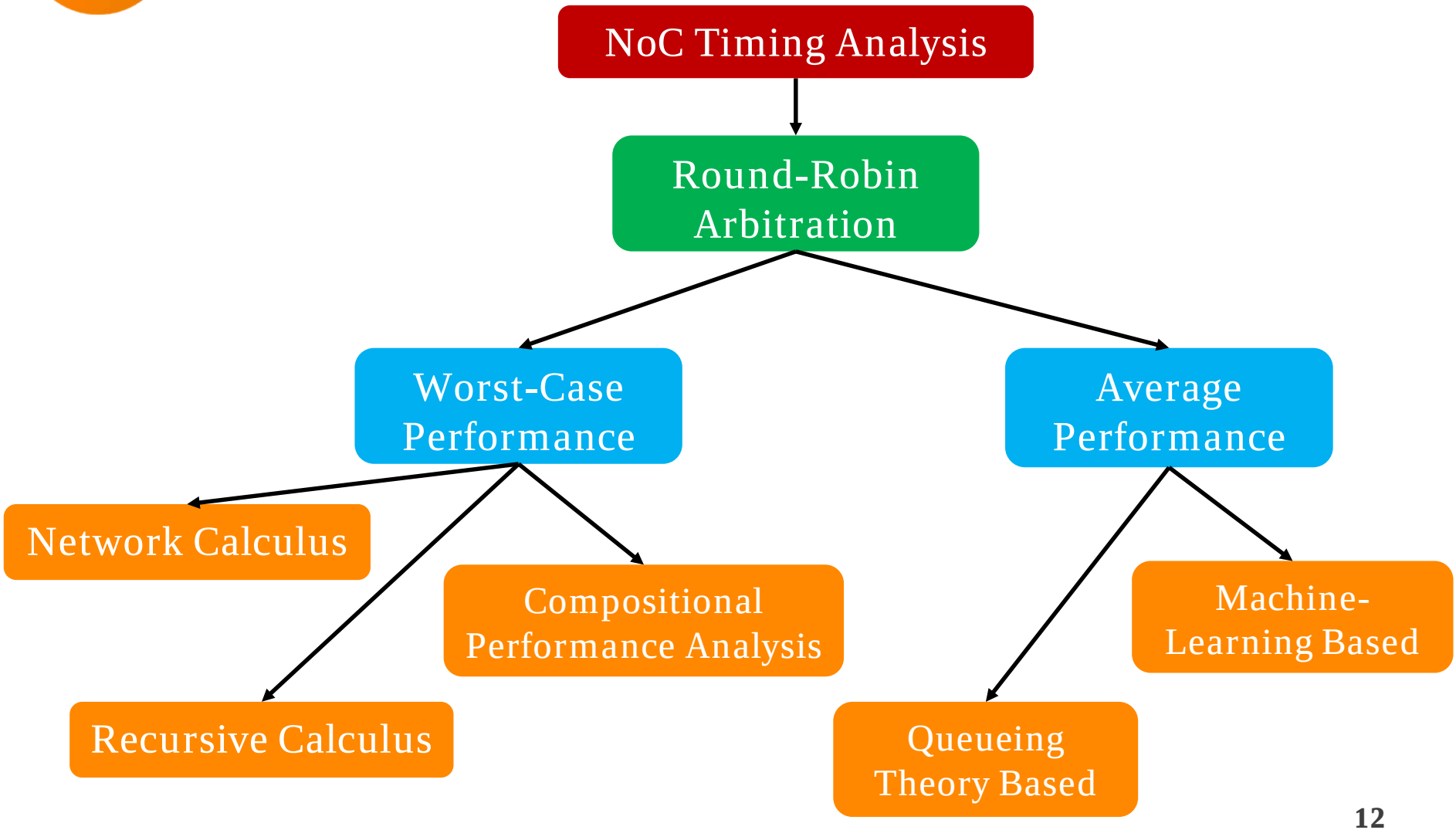
Motivation (2/3)



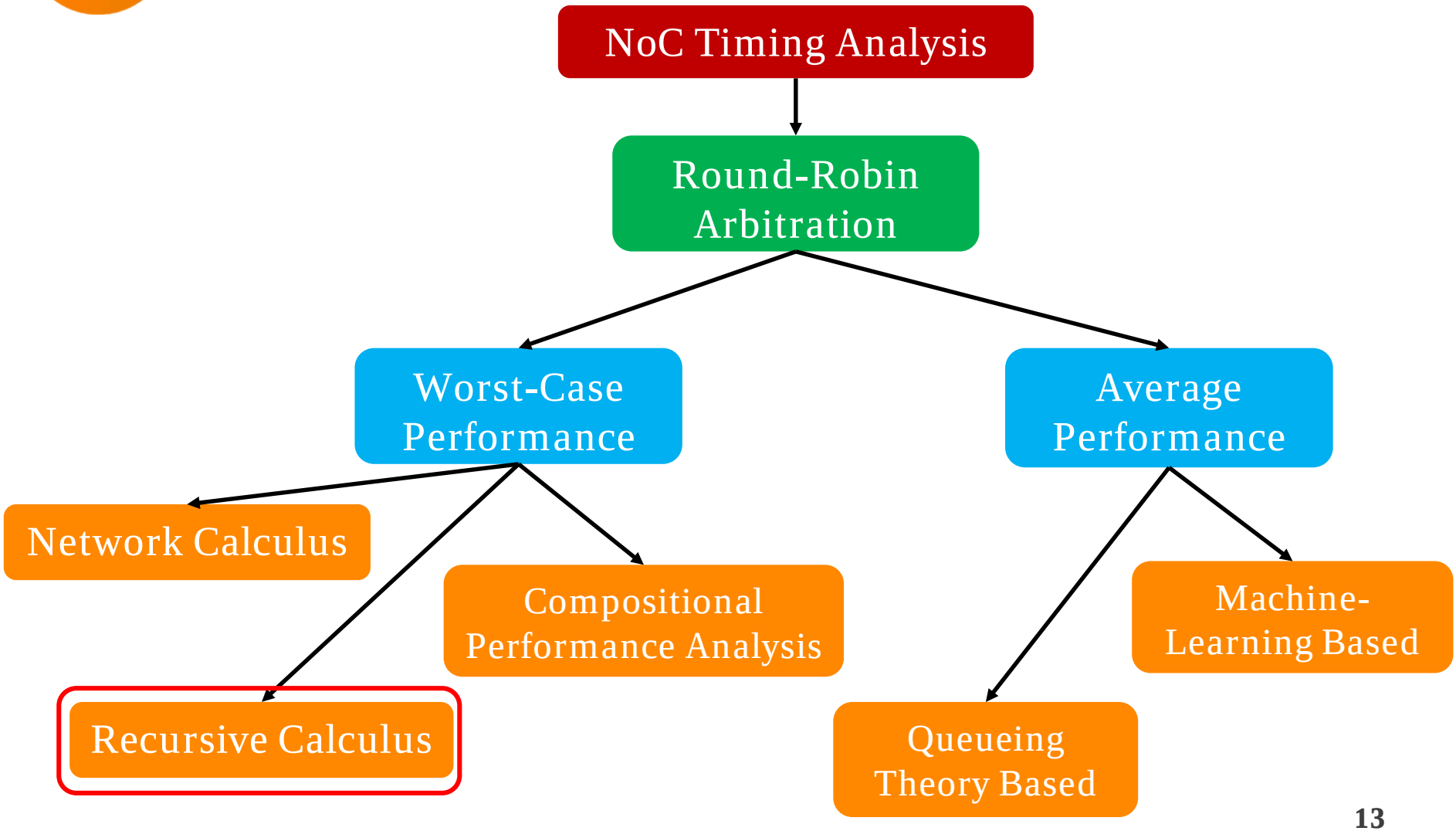
Motivation (2/3)



Motivation (2/3)



Motivation (2/3)



Motivation (3/3)

- Recursive Calculus (RC)
  - Computes Worst-Case Traversal Time (WCTT)
  - Pessimistic → no traffic patterns considered
  - Fast to compute

Motivation (3/3)

- Recursive Calculus (RC)
  - Computes Worst-Case Traversal Time (WCTT)
  - Pessimistic → no traffic patterns considered
  - Fast to compute

- Branch, Prune (BP) / Branch, Prune, Collapse (BPC)
  - Extension of RC
  - Less pessimistic → takes traffic pattern into account
  - High computational complexity

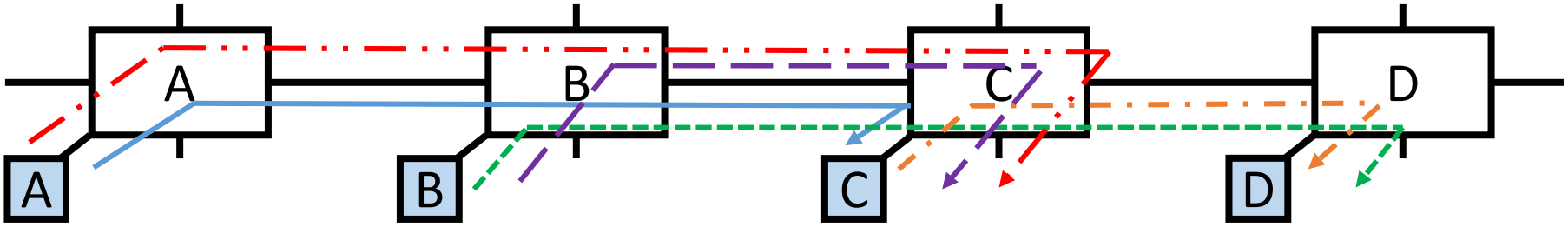
Motivation (3/3)

- Recursive Calculus (RC)
  - Computes Worst-Case Traversal Time (WCTT)
  - Pessimistic → no traffic patterns considered
  - Fast to compute

- Branch, Prune (BP) / Branch, Prune, Collapse (BPC)
  - Extension of RC
  - Less pessimistic → takes traffic pattern into account
  - High computational complexity

Tighter Recursive Calculus (TRC)
→ less pessimism than RC
→ lower computational complexity than BP/BPC

RC - Recap



$f_i = \{C_i, T_i, \Re_i\}$

f_1 = {3, 50, A → C}

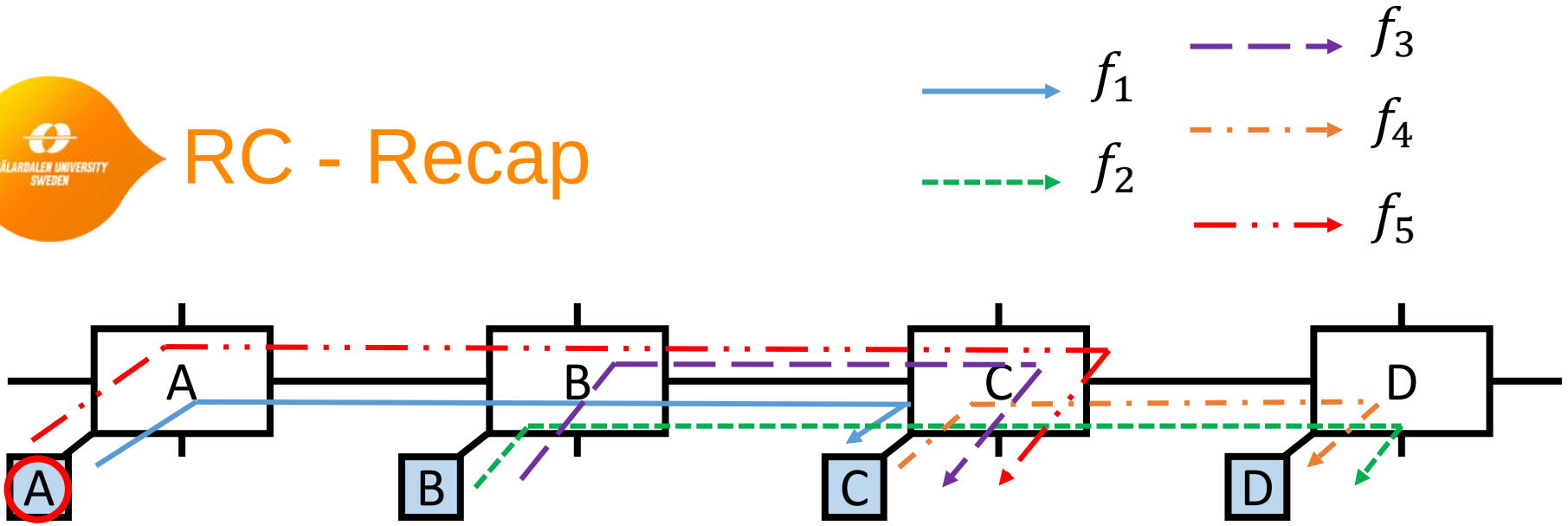
f_2 = {2, 40, B → D}

f_3 = {1, 20, B → C}

f_4 = {5, 20, C → D}

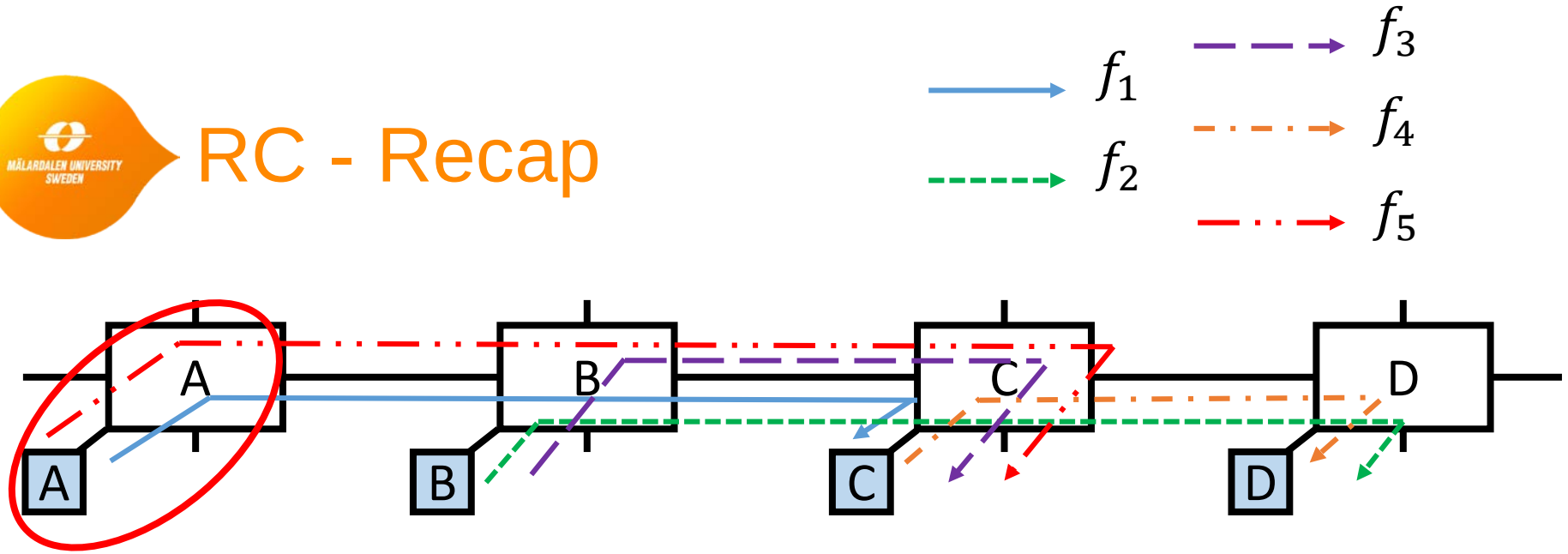
f_5 = {2, 20, A → C}

RC - Recap



w_1

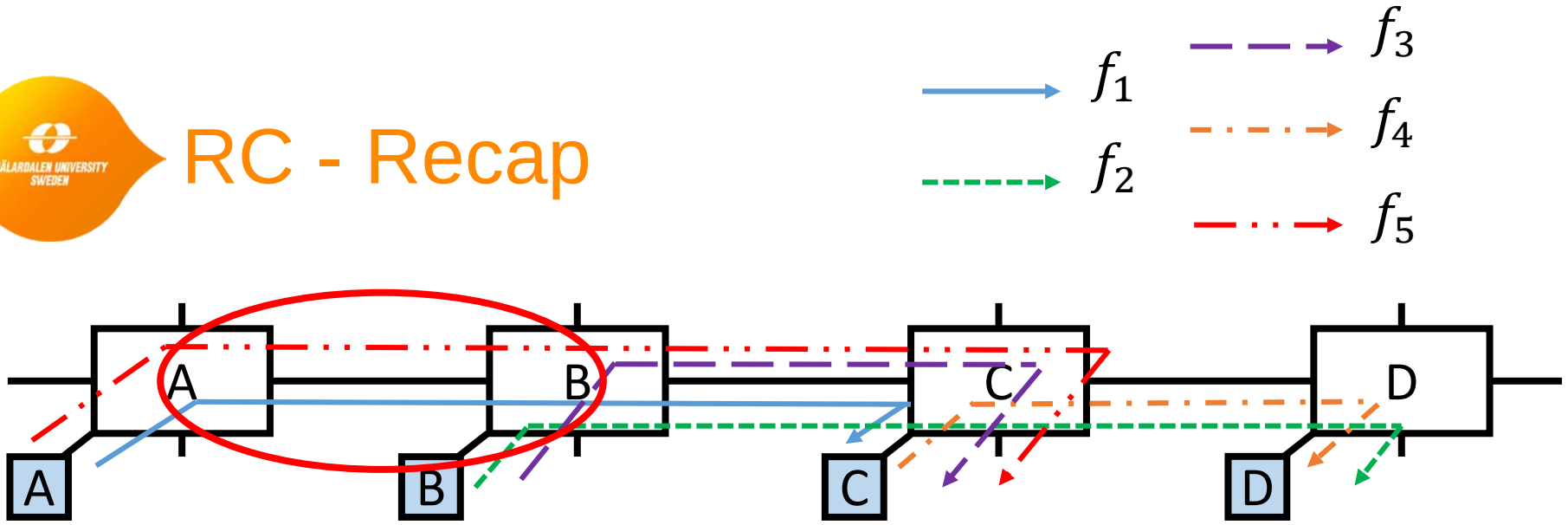
RC - Recap



$d(f_5, FirsLink)$

W_1

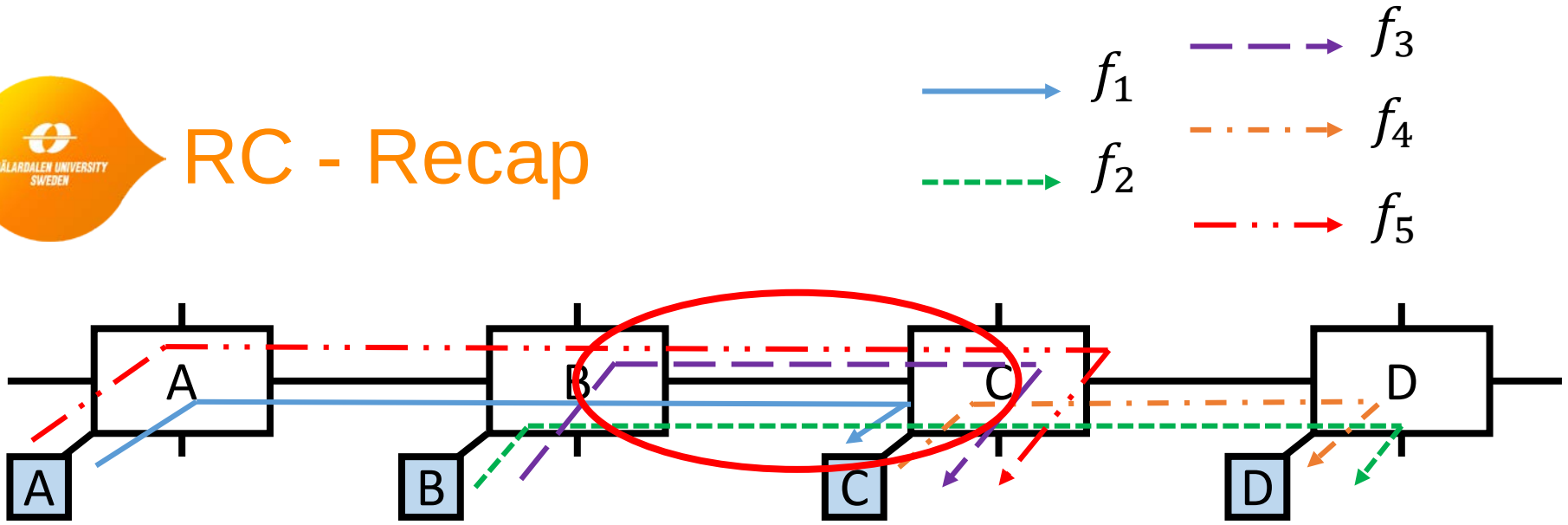
RC - Recap

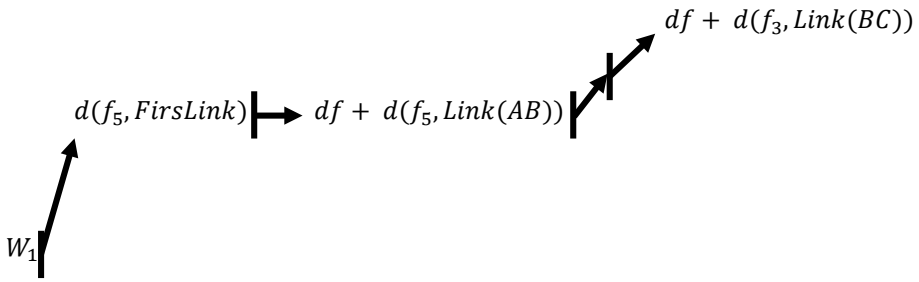


$d(f_5, FirsLink) \rightarrow df + d(f_5, Link(AB))$

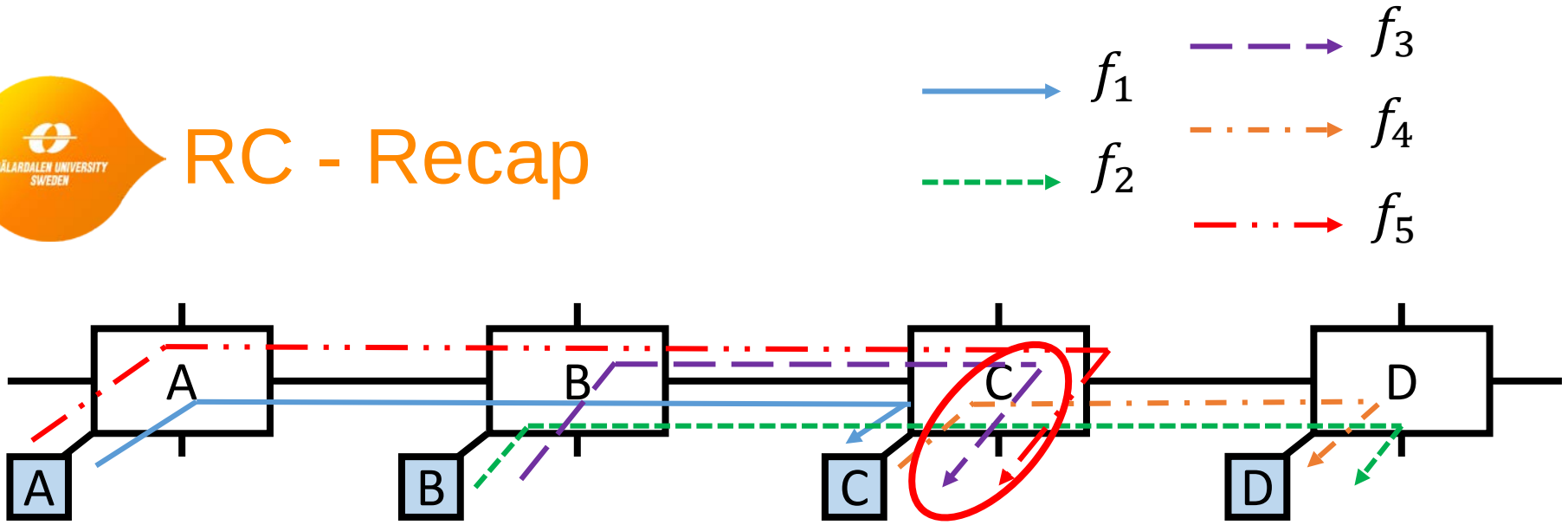
W_1

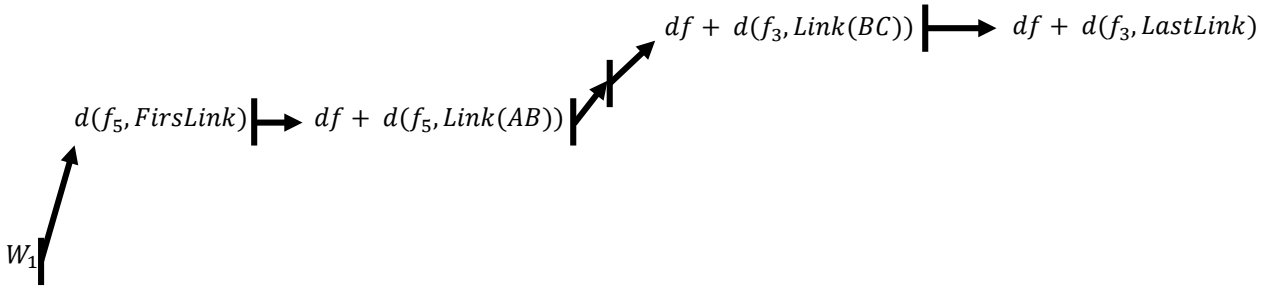
RC - Recap


f_1
f_2
f_3
f_4
f_5
A
B
C
D
A
B
C
D

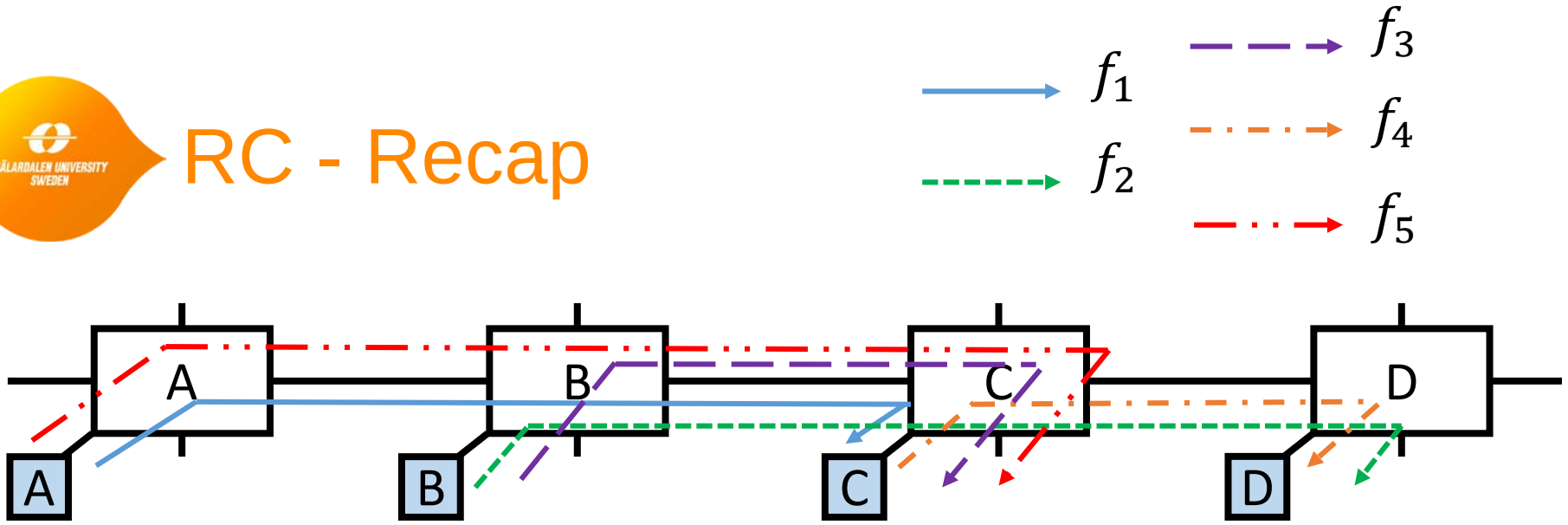

$df + d(f_3, Link(BC))$
$d(f_5, FirsLink)$
$df + d(f_5, Link(AB))$
W_1

RC - Recap


f_1
f_2
f_3
f_4
f_5
A
B
C
D
A
B
C
D


$df + d(f_3, Link(BC))$
$df + d(f_3, LastLink)$
$d(f_5, FirsLink)$
$df + d(f_5, Link(AB))$
W_1

RC - Recap

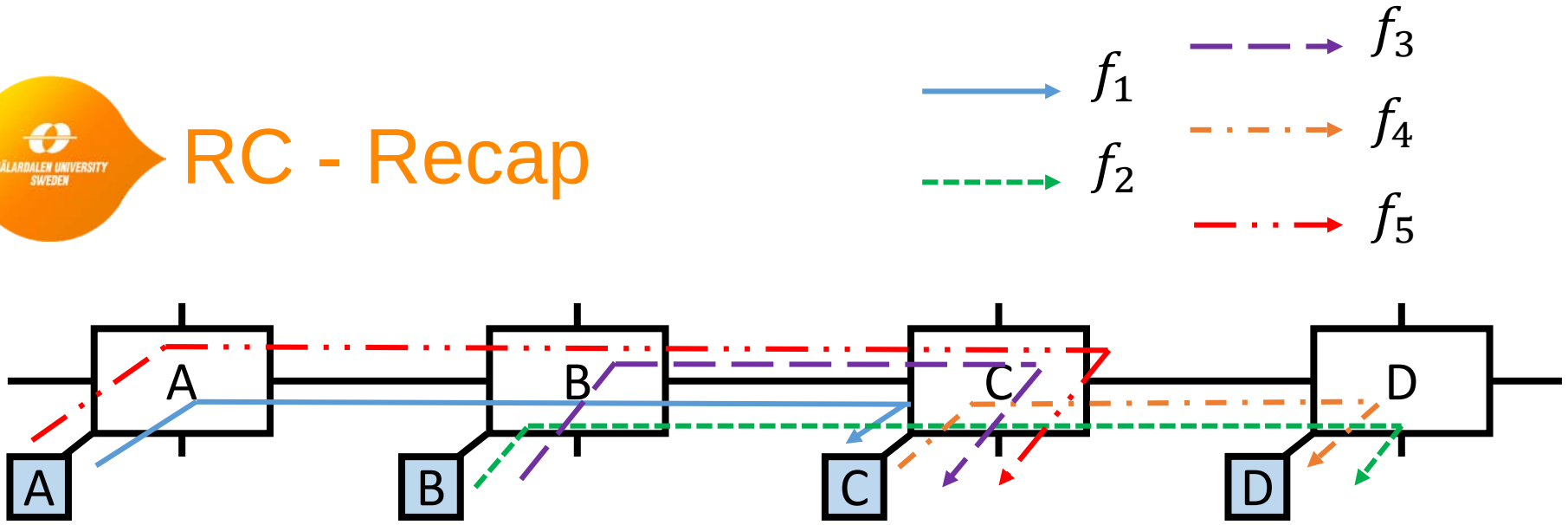


$df + d(f_3, Link(BC))$ → $df + d(f_3, LastLink)$ → L_3/bw

$d(f_5, FirsLink)$ → $df + d(f_5, Link(AB))$

W_1

RC - Recap



$df + d(f_3, Link(BC))$ → $df + d(f_3, LastLink)$ → L_3/bw

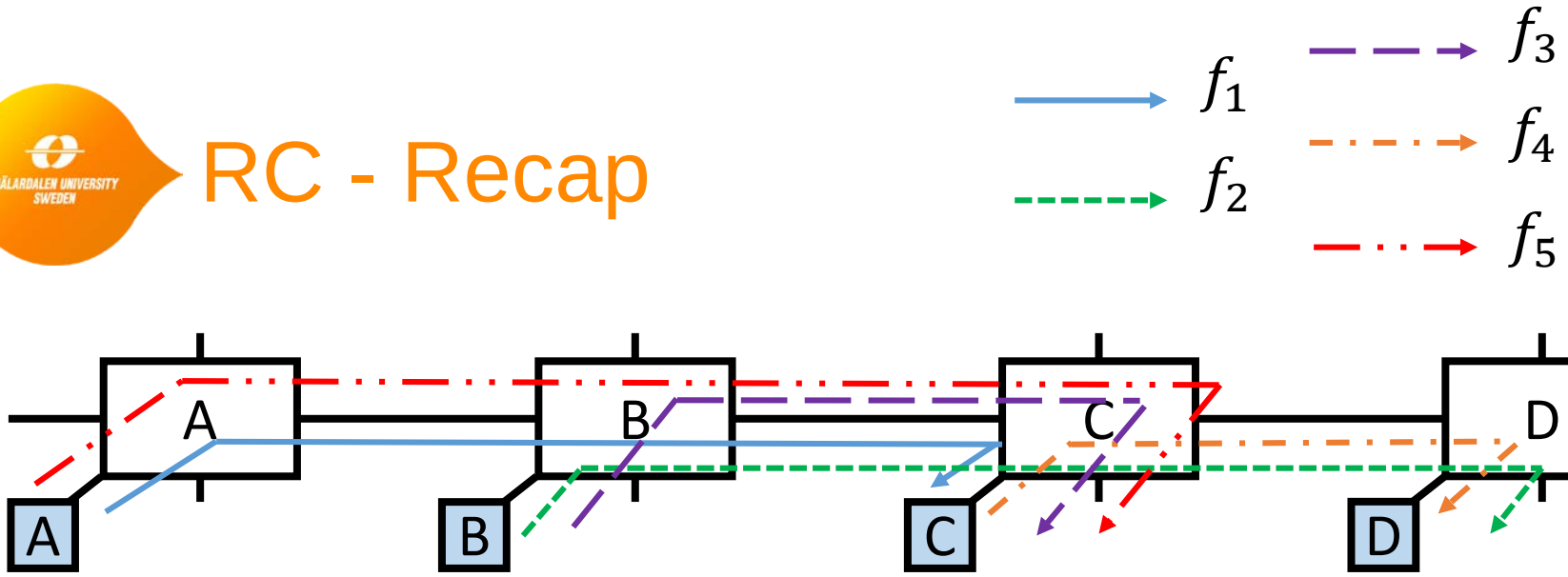
$d(f_5, FirsLink)$ → $df + d(f_5, Link(AB))$ → max → $df + d(f_2, Link(BC))$

$df + d(f_4, Link(CD))$ → $df + d(f_4, LastLink)$ → L_4/bw

$df + d(f_2, Link(CD))$ → $df + d(f_2, LastLink)$ → L_2/bw

W_1

RC - Recap

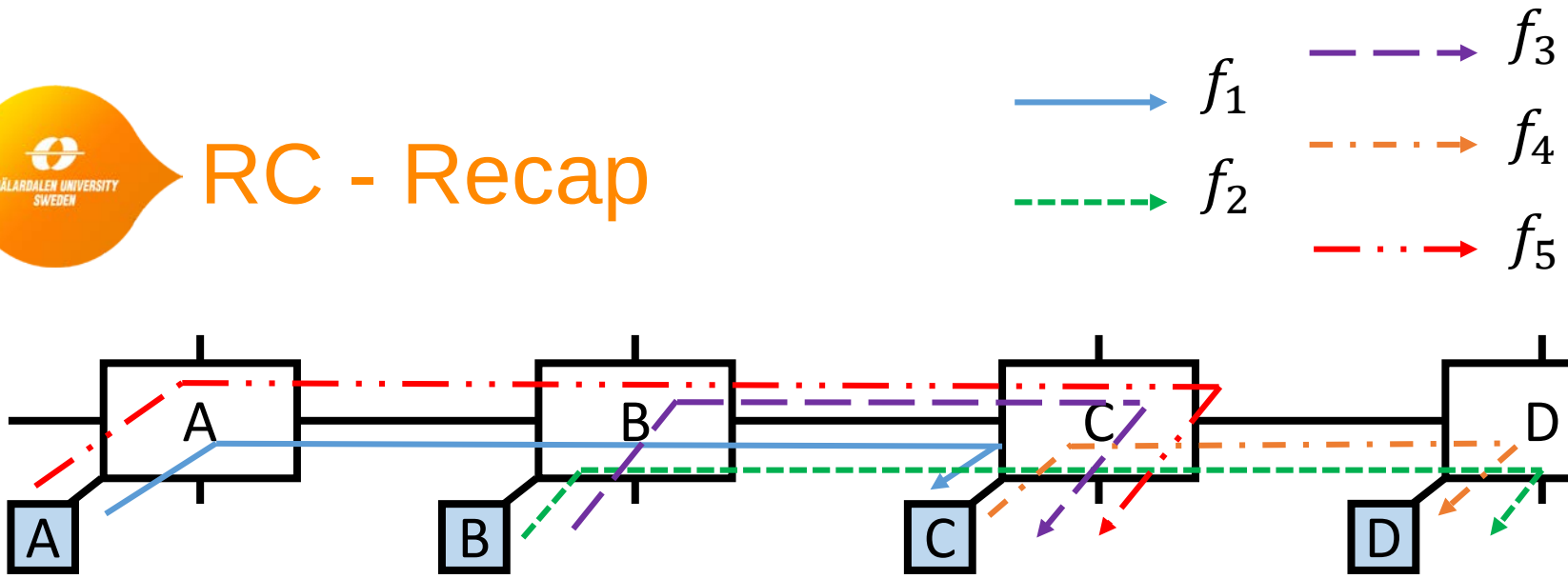


W_1

$d(f_5, FirsLink)$ → $df + d(f_5, Link(AB))$ → max(+)

- $df + d(f_3, Link(BC))$ → $df + d(f_3, LastLink)$ → L_3/bw
- $df + d(f_2, Link(BC))$ → (+)
  - $df + d(f_4, Link(CD))$ → $df + d(f_4, LastLink)$ → L_4/bw
  - $df + d(f_2, Link(CD))$ → $df + d(f_2, LastLink)$ → L_2/bw
- $df + d(f_5, Link(BC))$ → $df + d(f_5, LastLink)$ → L_5/bw

RC - Recap



W_1 +

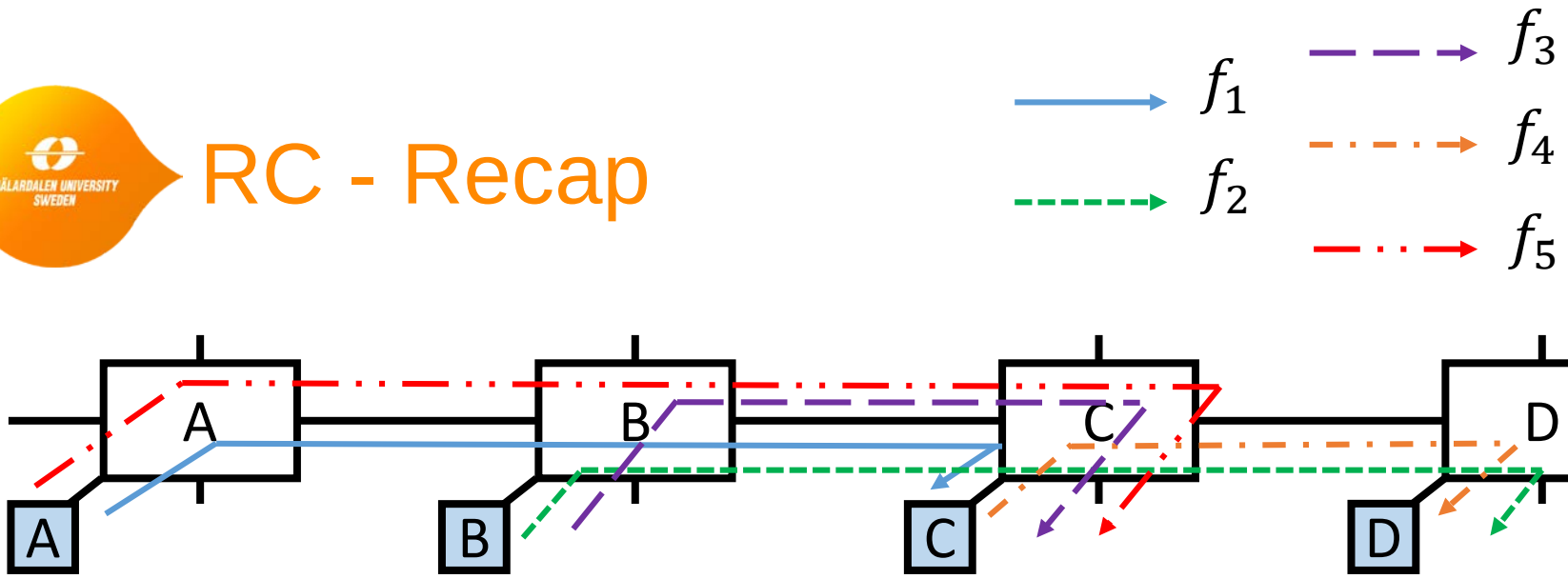
$d(f_5, FirsLink)$ → $df + d(f_5, Link(AB))$ → + → max

- $df + d(f_3, Link(BC))$ → $df + d(f_3, LastLink)$ → L_3/bw
- $df + d(f_2, Link(BC))$ → + → $df + d(f_4, Link(CD))$ → $df + d(f_4, LastLink)$ → L_4/bw
- $df + d(f_2, Link(BC))$ → + → $df + d(f_2, Link(CD))$ → $df + d(f_2, LastLink)$ → L_2/bw
- $df + d(f_5, Link(BC))$ → $df + d(f_5, LastLink)$ → L_5/bw

$d(f_1, FirsLink)$ → $df + d(f_1, Link(AB))$ → + → max

- $df + d(f_3, Link(BC))$ → $df + d(f_3, LastLink)$ → L_3/bw
- $df + d(f_2, Link(BC))$ → + → $df + d(f_4, Link(CD))$ → $df + d(f_4, LastLink)$ → L_4/bw
- $df + d(f_2, Link(BC))$ → + → $df + d(f_2, Link(CD))$ → $df + d(f_2, LastLink)$ → L_2/bw
- $df + d(f_1, Link(BC))$ → $df + d(f_1, LastLink)$ → L_1/bw

RC - Recap



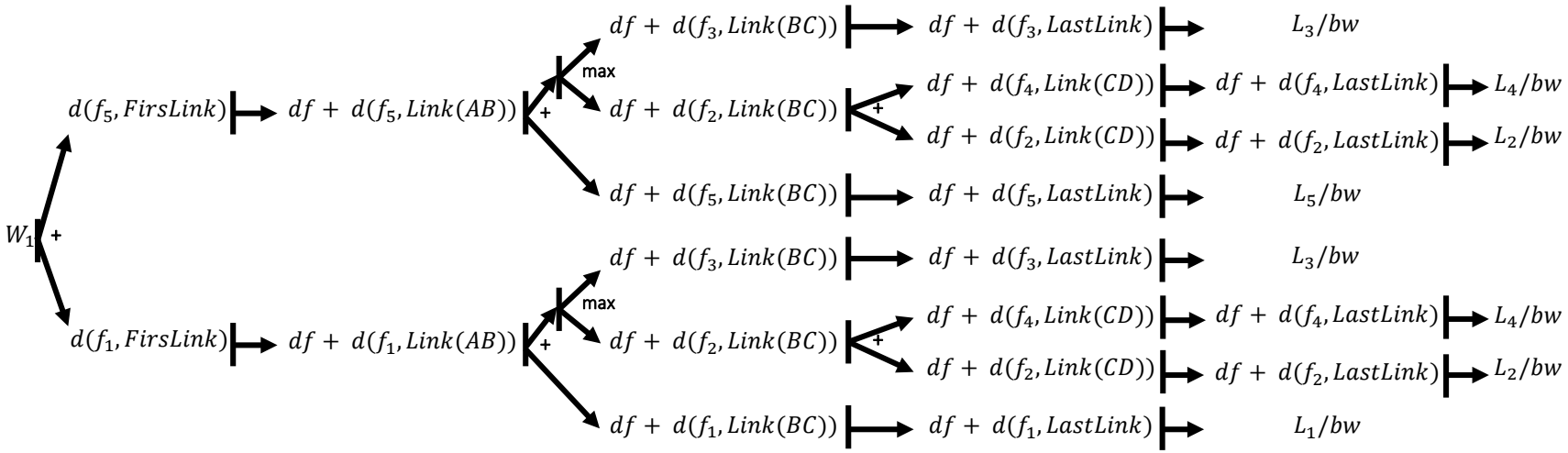
W_1 +

- $d(f_5, FirsLink) \rightarrow df + d(f_5, Link(AB))$ +
  - max
    - $df + d(f_3, Link(BC)) \rightarrow df + d(f_3, LastLink) \rightarrow L_3/bw$
    - $df + d(f_2, Link(BC))$ +
      - $df + d(f_4, Link(CD)) \rightarrow df + d(f_4, LastLink) \rightarrow L_4/bw$
      - $df + d(f_2, Link(CD)) \rightarrow df + d(f_2, LastLink) \rightarrow L_2/bw$
  - $df + d(f_5, Link(BC)) \rightarrow df + d(f_5, LastLink) \rightarrow L_5/bw$
- $d(f_1, FirsLink) \rightarrow df + d(f_1, Link(AB))$ +
  - max
    - $df + d(f_3, Link(BC)) \rightarrow df + d(f_3, LastLink) \rightarrow L_3/bw$
    - $df + d(f_2, Link(BC))$ +
      - $df + d(f_4, Link(CD)) \rightarrow df + d(f_4, LastLink) \rightarrow L_4/bw$
      - $df + d(f_2, Link(CD)) \rightarrow df + d(f_2, LastLink) \rightarrow L_2/bw$
  - $df + d(f_1, Link(BC)) \rightarrow df + d(f_1, LastLink) \rightarrow L_1/bw$

$$W_1 = 33$$

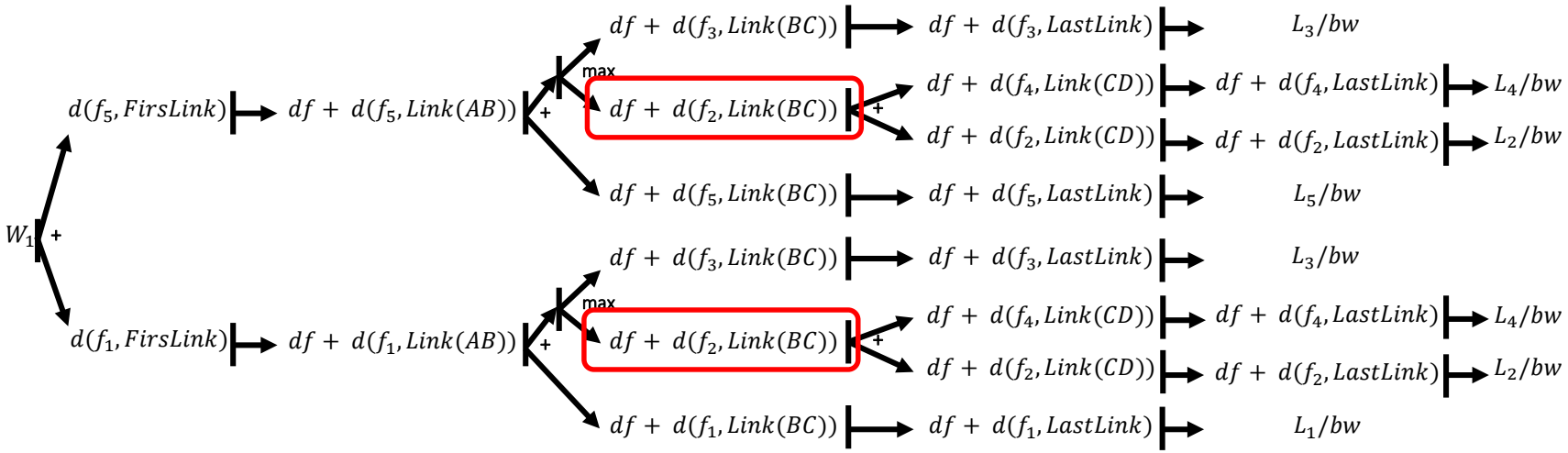
Pessimism in RC

- Arrival patterns are not taken into account
  - All flows are always assumed to be pending



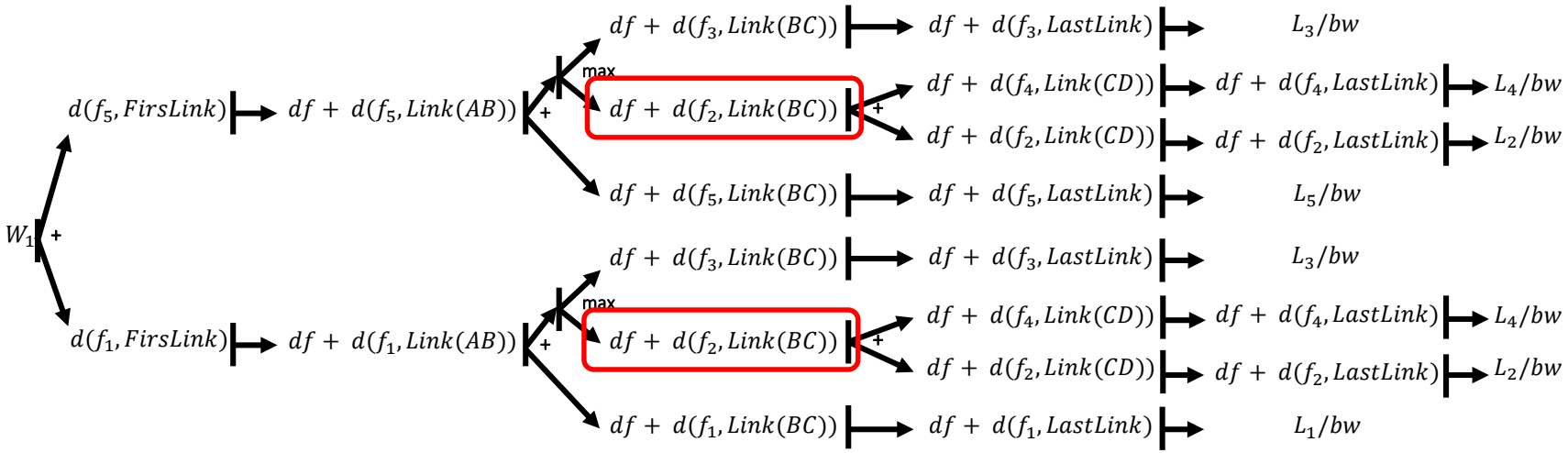
Pessimism in RC

- Arrival patterns are not taken into account
  - All flows are always assumed to be pending



Pessimism in RC

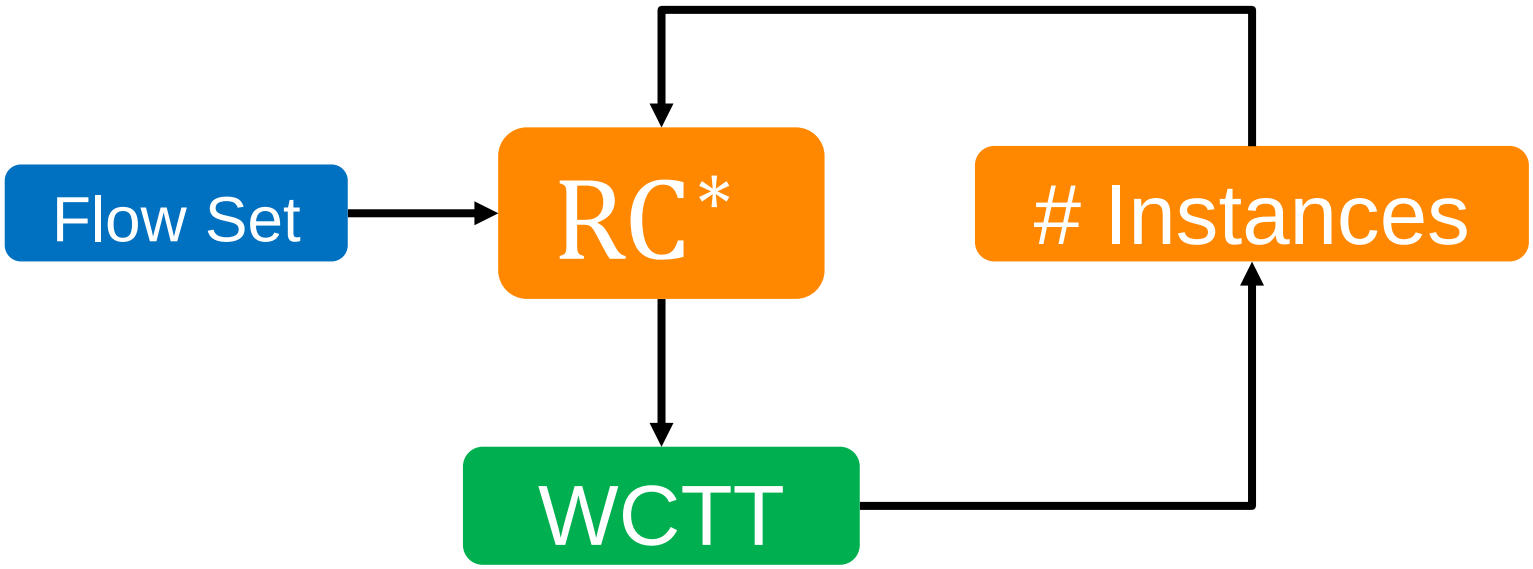
- Arrival patterns are not taken into account
  - All flows are always assumed to be pending



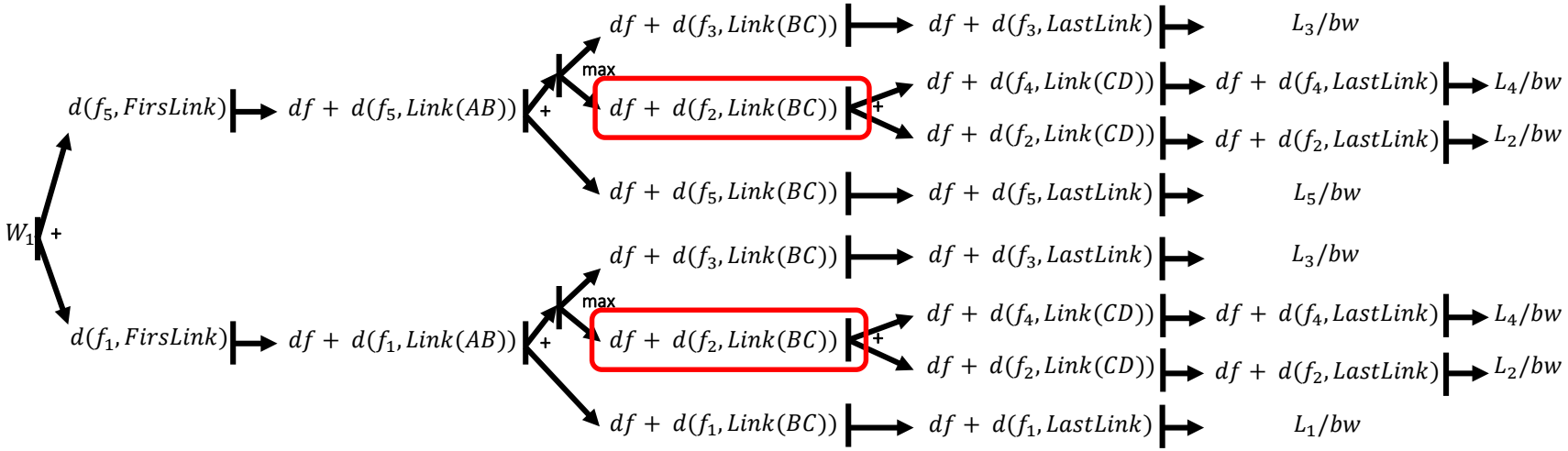
$W_1 = 33$ but $T_2 = 40$ → Only one instance of f_2 can occur

Tighter Recursive Calculus

- Arrival patterns are taken into account

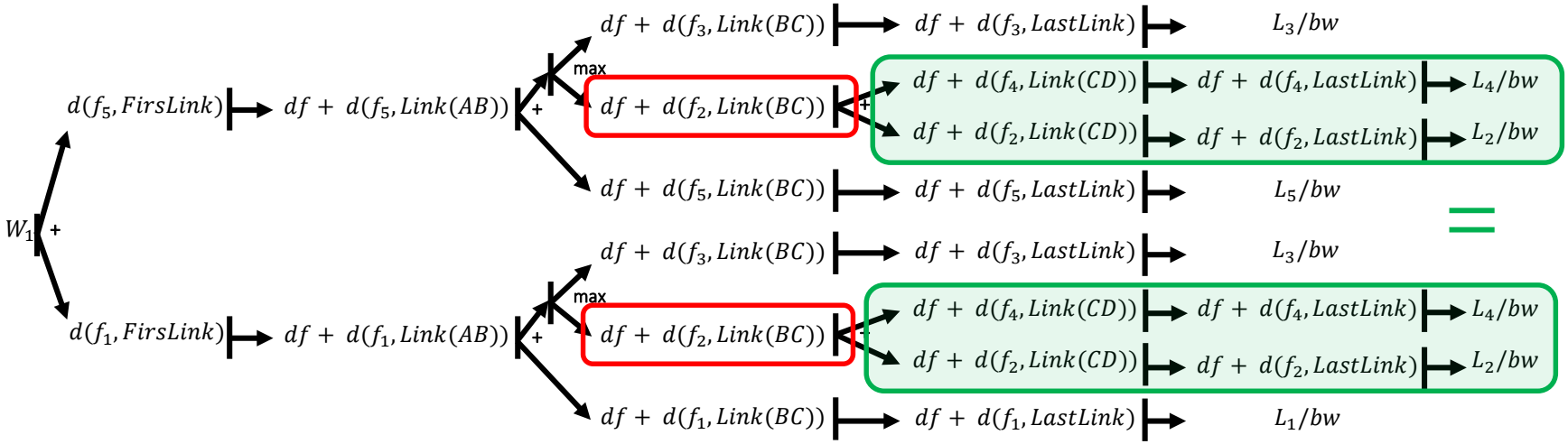


Pruning the Computation Tree



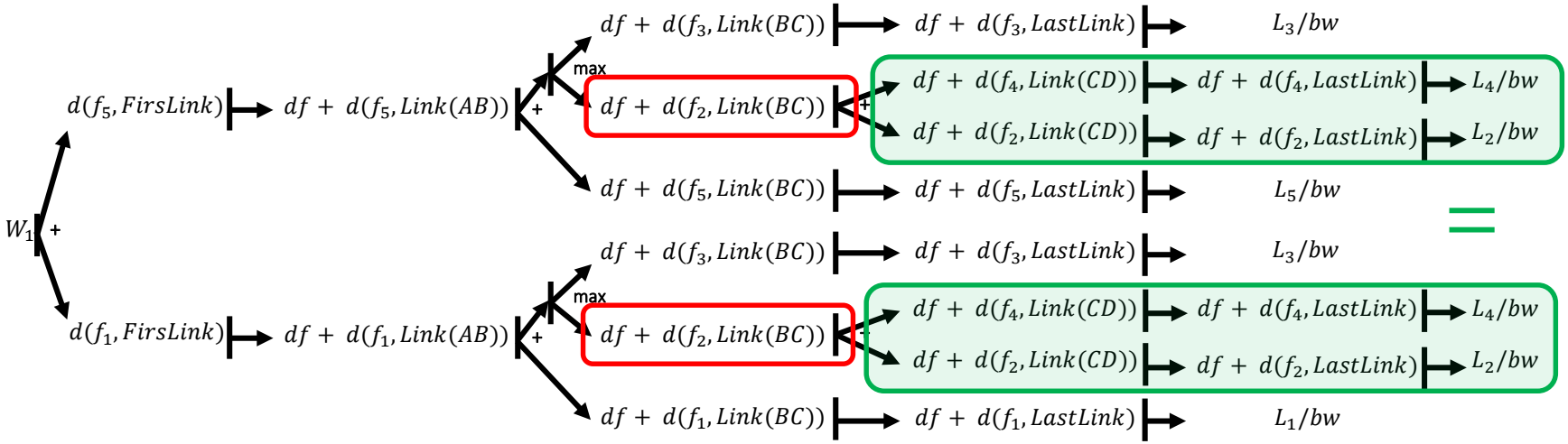
- Only one instance of f_2 possible
  - Which one to cut?

Pruning the Computation Tree



- Only one instance of f_2 possible
  - Which one to cut?

Pruning the Computation Tree

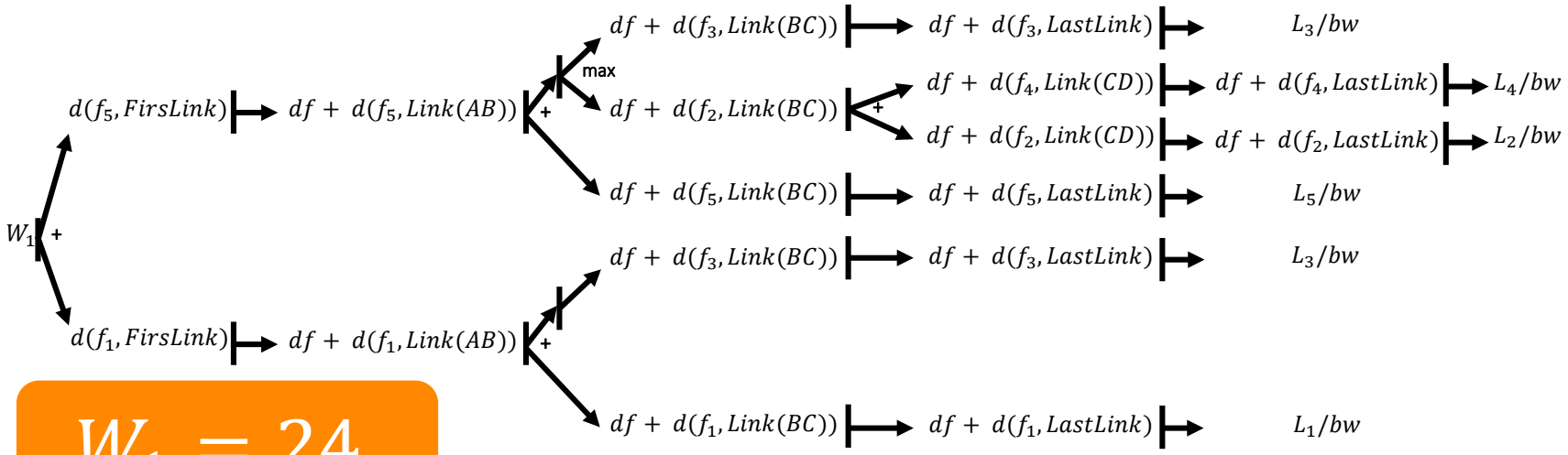


- Only one instance of f_2 possible
  - Which one to cut?

Formal Proof to support this observation in the paper!

Modification on RC

- A list is added, containing the number of occurrences allowed for each flow
- While traversing the computation tree, the possible remaining instances of the flows are monitored
- If remaining instance becomes 0 → no more blocking from this flow is considered

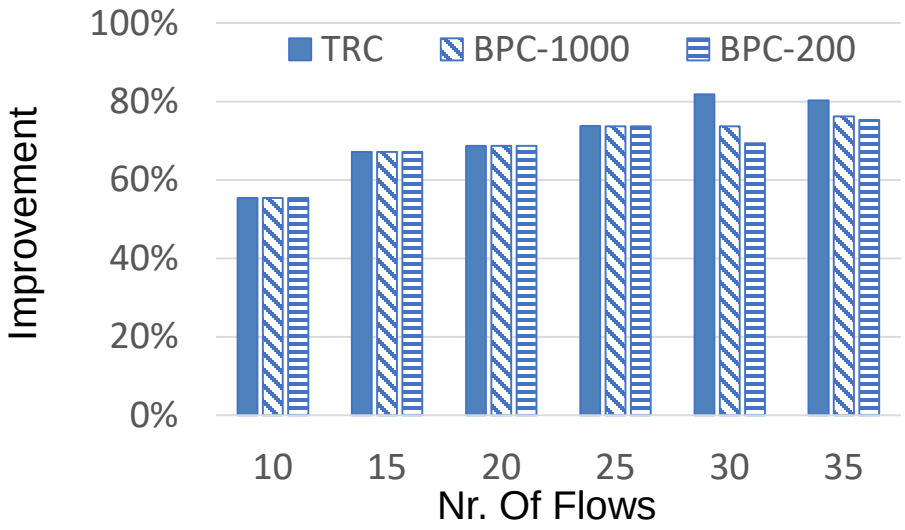


$$W_1 = 24$$

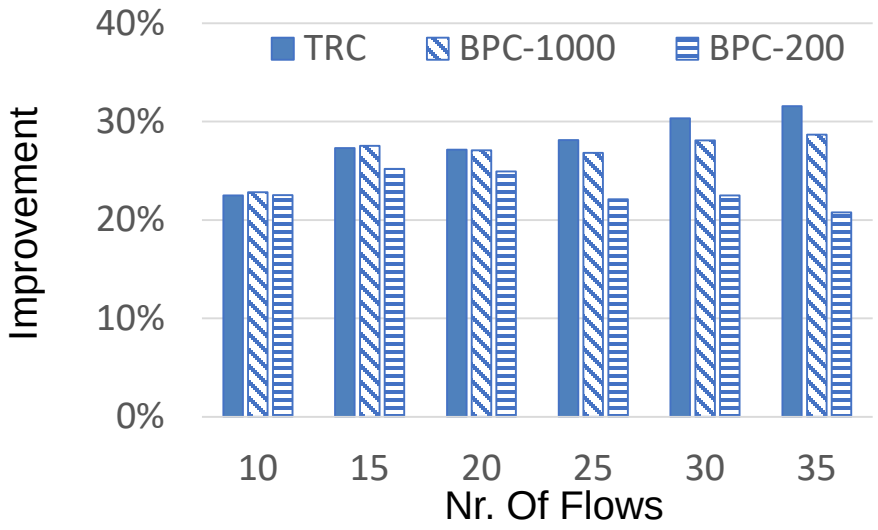
Evaluation

- 4 × 4 2D-meshed NoC
  - 100 MHz and 3.2 Gbit/s.
- Randomly generated flows
  - Following uniform distribution
  - Period – [5,33] ms
  - Utilization – [0.003, 0.1]
- We compare TRC with RC, and BPC
  - Accuracy of the results
  - Computation time

Evaluation – Accuracy of Results

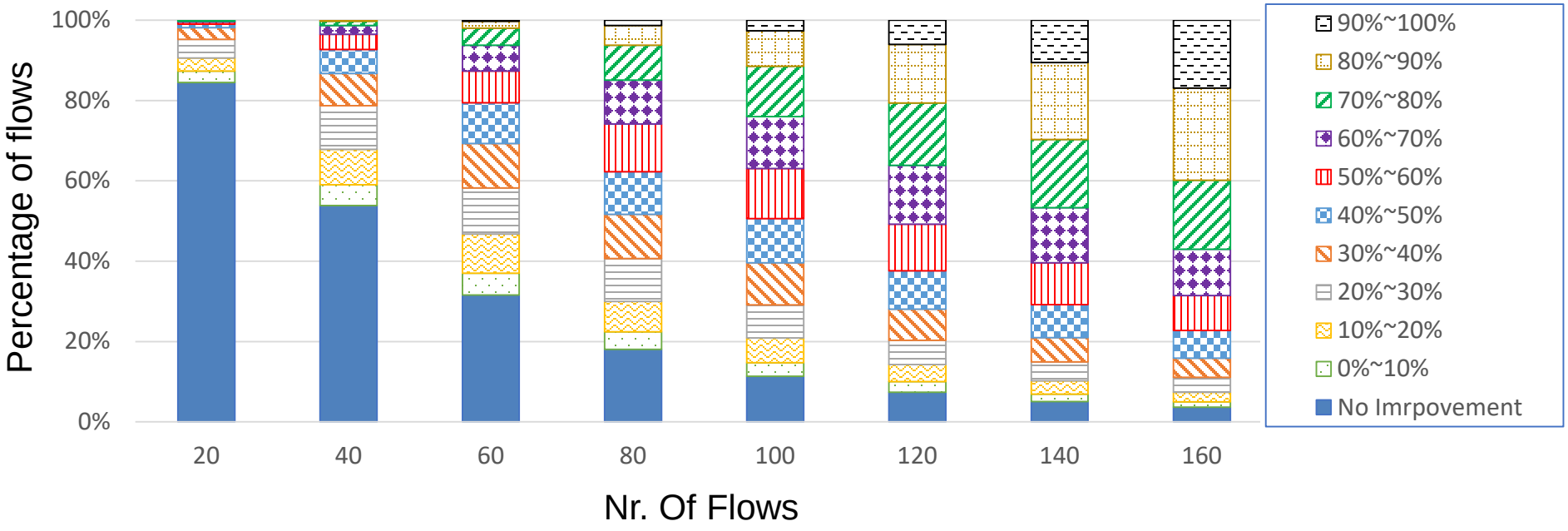


(a) Maximum Improvement

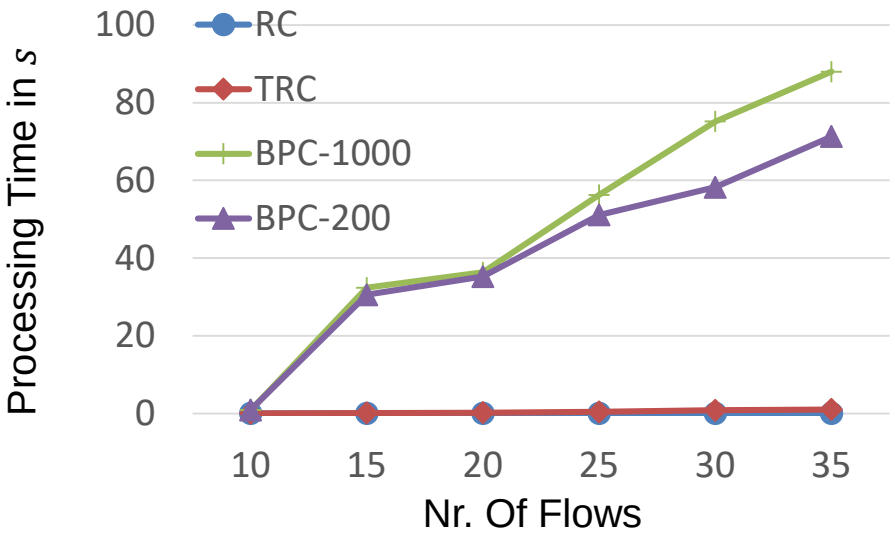


(b) Average Improvement

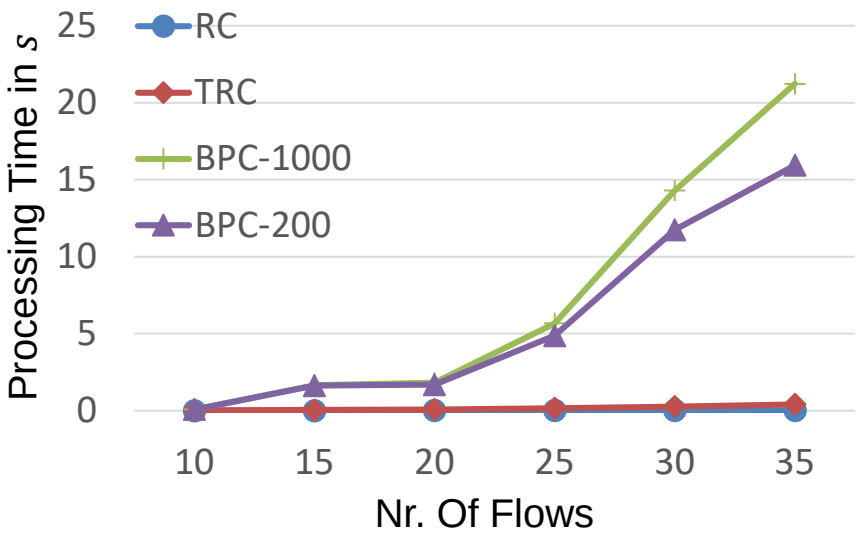
Evaluation - Improvement



Evaluation – Computation Time

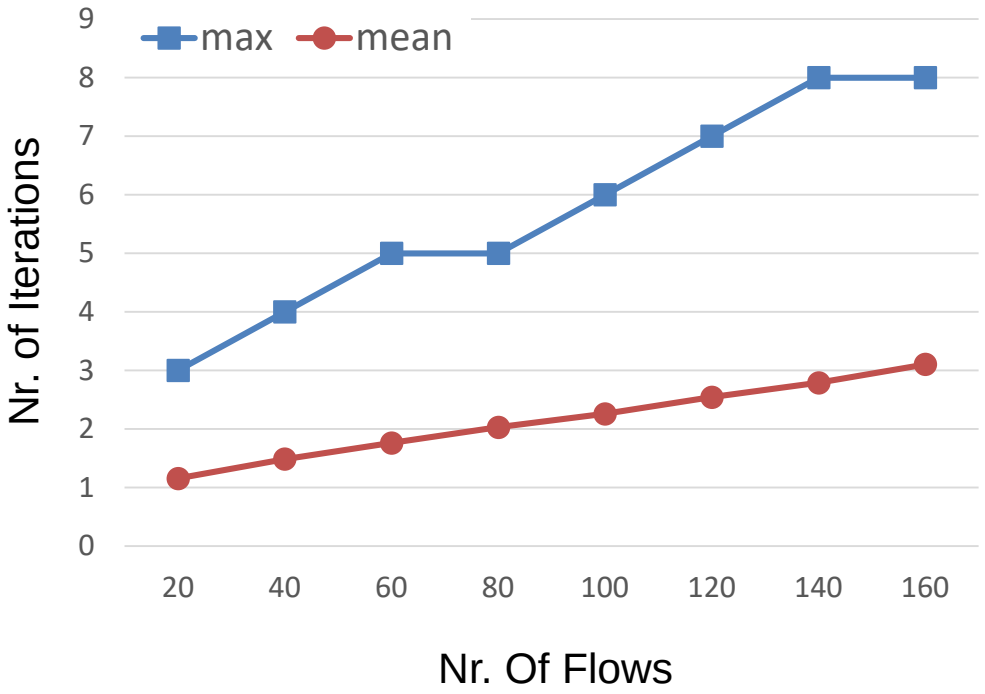


(a) Max. processing time per flow set



(b) Avg. processing time per flow set

Evaluation – Iterations



Conclusions

- ✓ We present an improved Recursive Calculus that decreases pessimism while maintaining a low computational complexity
- ✓ Extensive evaluations are performed to support the claims

Conclusions

- ✓ We present an improved Recursive Calculus that decreases pessimism while maintaining a low computational complexity
- ✓ Extensive evaluations are performed to support the claims

  - Extend the RC to incorporate buffer sizes on the NoC routers
  - Perform a general comparison between RC and other methods

Thank you for the attention!

Questions?