Upstream Progressive Reconfiguration

Juan José García-Castro Crespo

April 2020

Contents

Chapter 1

Working example

This section shows a working example of UPR applied on a 3x3 mesh topology with one source terminal $T0$ attached to switch $S0$, and two sink terminals $T7$ and $T8$ attached to switch $S7$ and $S8$ respectively. Figure 1.1 shows the proposed scenario. Initially, routes follow the *Dimension Order XY (xy)* routing algorithm. The final routing algorithm to be applied after the reconfiguration process is the *Odd-even (oe)* routing algorithm.

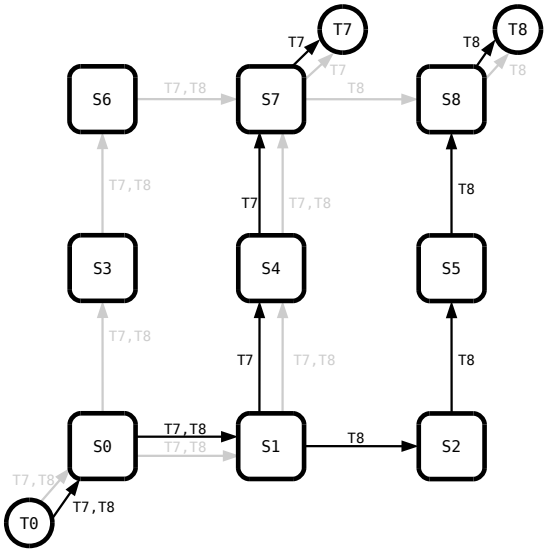
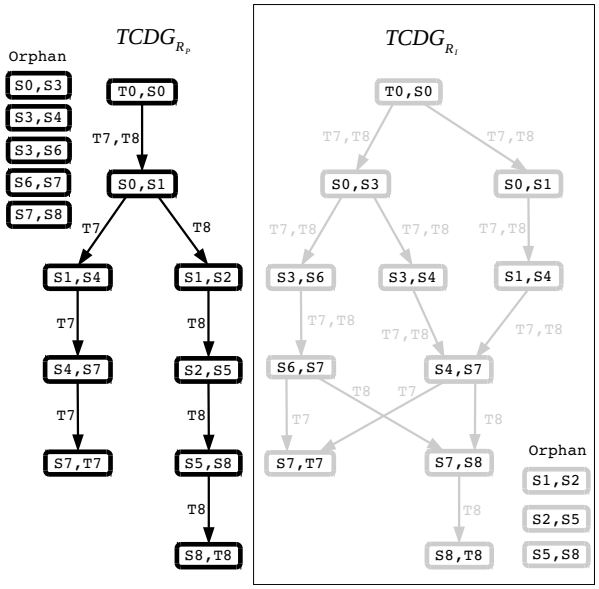
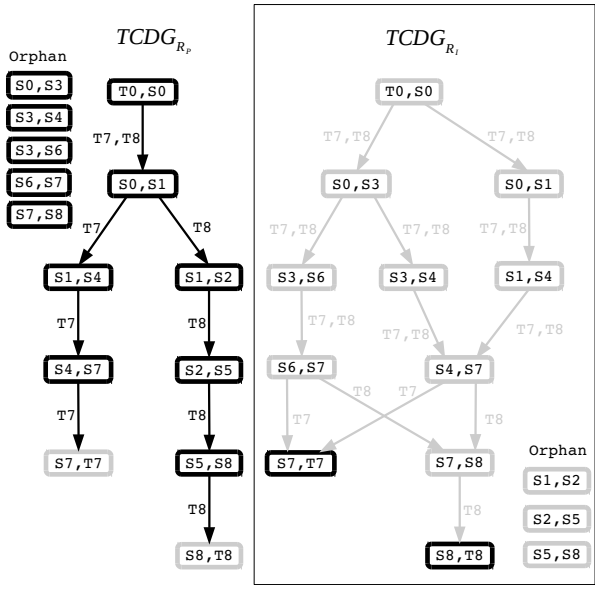


Figure 1.1: UPR working example on a 3x3 mesh with three terminal nodes involved. Source terminal $T0$, and destination terminals $T7$ and $T8$. Initial/final routing algorithms are *xy* and *oe* respectively. Paths from $T0$ towards $\{T7, T8\}$ under *xy* (black) and *oe* (gray).

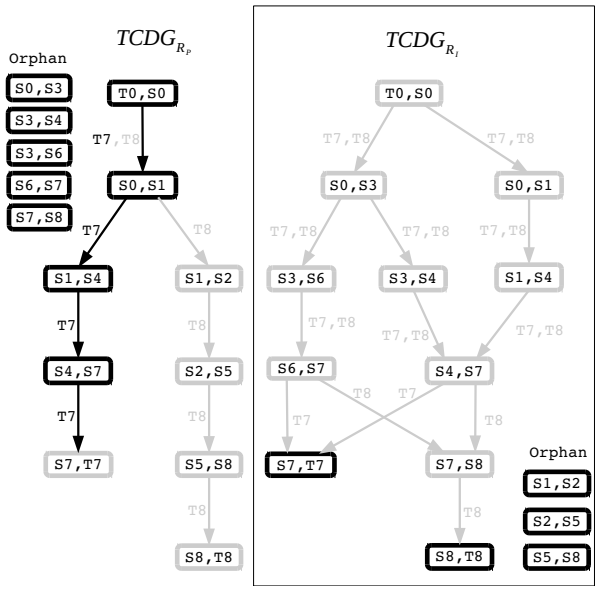
Figures 1.2 and 1.3 give a detailed tracing of the reconfiguration process step by step. In these figures, graph vertices represent channels within the network, whilst graph edges represent dependencies among channels. Besides, black colored channels indicate that those channels are using the routing choices provided by the routing function within which the black colored channel is contained. On the contrary, gray colored channels mean that they are not applying routing choices under that routing function. Channels shall be applying routing choices under a single routing function.



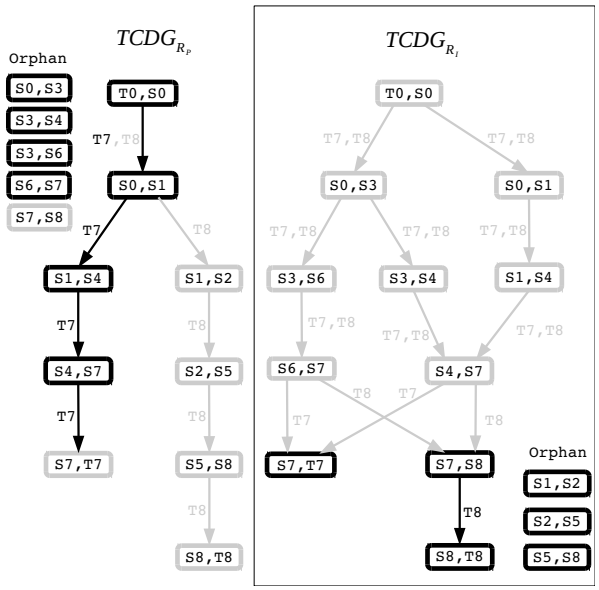
(a) Initial state. All channels are using the routing choices provided by R_P (xy in this case).



(b) Channels $(S7, T7)$ and $(S8, T8)$ upgrade to R_I, they fulfill $cond_i$ because they are sink channels.

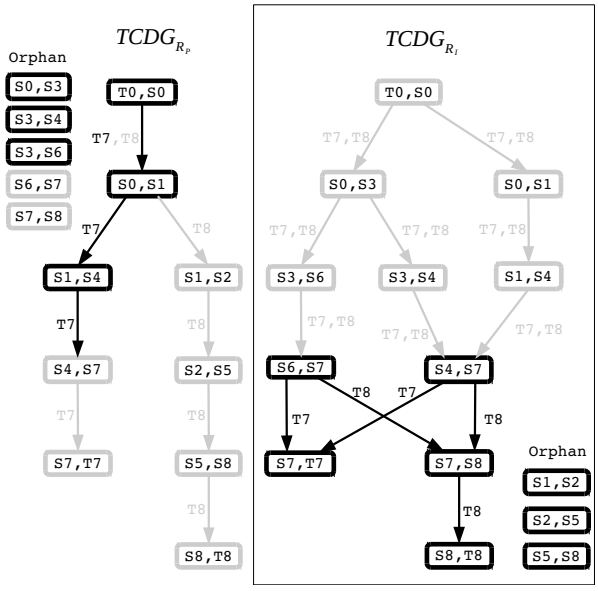


(c) Orphan channels $(S1, S2)$, $(S2, S5)$ and $(S5, S8)$ in R_I request removal of dependencies bringing target $T8$ at R_P. Hence, removal requests propagate upwards following R_P to $(T0, S0)$, which halts injection of target $T8$. Once target $T8$ is drained, orphan channels in R_I upgrade.

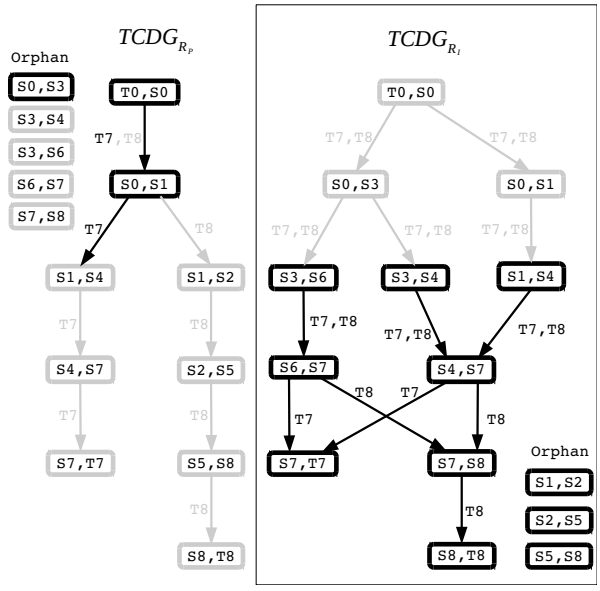


(d) Following the reverse topological order, channel $(S7, S8)$ evaluates $cond_i$ which is satisfied because no targets are brought into $(S7, S8)$ at R_P (it is an orphan channel in R_P). Therefore it upgrades to R_I.

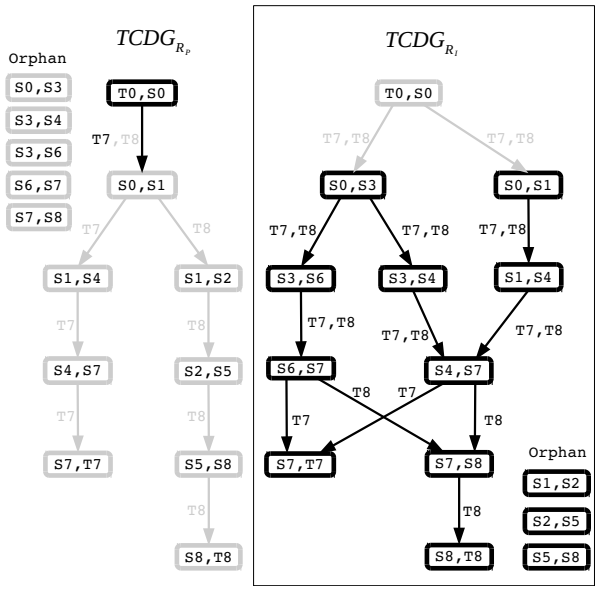
Figure 1.2: UPR working example on a 3x3 mesh with three terminal nodes involved. Source terminal $T0$, and destination terminals $T7$ and $T8$. Initial/final routing algorithms are xy and oe respectively.



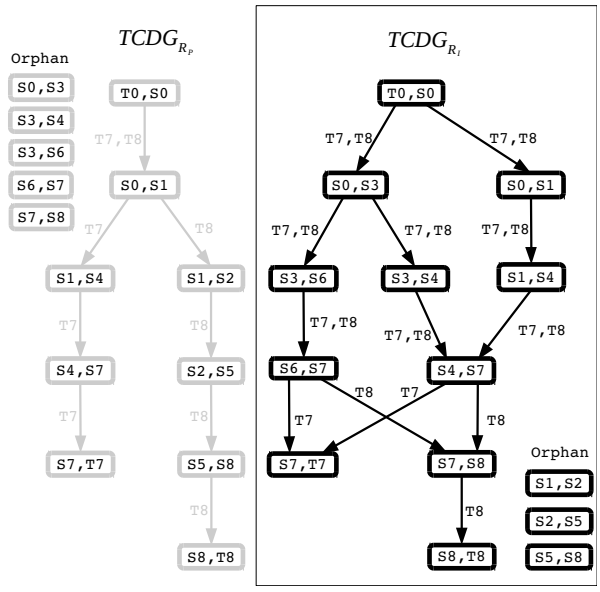
(a) Channels $(S6, S7)$ and $(S4, S7)$ upgrade to R_I. $(S6, S7)$ was an orphan channel under R_P. On the other hand, $(S4, S7)$ fulfills $cond_i$ because it provides an alternative route for target $T7$ in R_I.



(b) Channels $(S3, S6)$, $(S3, S4)$ and $(S1, S4)$ upgrade to R_I. Channels $(S3, S6)$ and $(S3, S4)$ fulfill $cond_i$ because they were orphan channels under R_P. Meanwhile, $(S1, S4)$ provides an alternative route for target $T7$ in R_I.



(c) Channels $(S0, S3)$ and $(S0, S1)$ upgrade to R_I.



(d) Final state. Source channel $(T0, S0)$ upgrades to R_I.

Figure 1.3: UPR working example on a 3x3 mesh with three terminal nodes involved. Source terminal $T0$, and destination terminals $T7$ and $T8$. Initial/final routing algorithms are *xy* and *oe* respectively.

Chapter 2

Evaluation and results

In this section, we evaluate the proposed reconfiguration scheme. The goal of this evaluation is to assess the suitability of UPR and to show that it can benefit from compatibility exploitation between two routing algorithms to reduce *selective halting* at sources while also reducing the need for packet drainage at channels.

In the first part, we describe the methodology, topology and routing algorithms used for the evaluation. Afterwards, we explain the metrics used to present the results and analysis.

2.1 Methodology

For our proposal evaluation we developed a process-based discrete-event simulation tool written in the *Python* programming language which allowed us to represent the reconfiguration process as a concurrent computation of graph operations by multiple processes. The core of our simulation tool uses the *SimPy*[**?**] framework to model an asynchronous event dispatcher.

In this tool, channel dependency data is available globally for all channels. This information is stored using graph representations[1] of $TCDG_{R_S}$, $TCDG_{R_P}$, $TCDG_{R_I}$ and $TCDG_{R_F}$ associated to each routing function. These shared data structures yield a critical section and they are accessed by channels to read and/or update their dependencies information atomically. This guarantees dependency information to be updated at all times so that decisions made by channels are always based on the current state of the system.

In the concurrent model included in the simulation, each process comprises operations made by a single unidirectional channel (channel for short) within the directed graph representing the network topology. Each channel c is in charge of the addition/removal of outgoing dependencies devised from its local perspective. Therefore, c will only perform modifications to $TCDG_{R_P}$, $TCDG_{R_I}$ involving dependencies in $TCDG_{Local_{R_P\{c\}}}$, $TCDG_{Local_{R_I\{c\}}}$. In other words, dependencies within sets $D^+_{TCDG_{R_P}}(c)$ and $D^+_{TCDG_{R_I}}(c)$.

Finally, channels' actions are triggered by means of messages exchanged between neighboring channels. These messages contain the necessary information to notify about a particular event/operation performed by the sender channel.

2.1.1 Topology

The network is built from a 5×5 mesh topology using bidirectional physical links between router nodes. Each bidirectional link is considered as two independent unidirectional channels. For simplicity, we assume a single virtual channel (i.e. no virtual channels).

2.1.2 Routing algorithms

We consider four different minimal routing algorithms: non-adaptive algorithms *xy* and *yx*, and the partially adaptive algorithms *odd-even*[2] and *negative-first*[3]. Routing algorithms *xy* and *yx* lack of any degree of adaptiveness, providing a single route per source-destination pair. On the other hand, *odd-even* (*oe*) and *negative-first* (*nf*) may provide multiple routes for each source-destination pair and a reasonable degree of adaptiveness.

Each combination of two distinct routing algorithms results in a different scenario for the reconfiguration process to change from one routing algorithm to the other.

[1] Provided by the *NetworkX*[**?**] python library

2.1.3 UPR operations considered

For this evaluation, two main scenarios are considered regarding the UPR reconfiguration scheme. In the first scenario, a minimal UPR version has been configured. That is, relying on *selective halting* only at channels.

A second scenario is considered, where all possible graph manipulation operations are perfomed by channels during the reconfiguration process. This includes conformability and compatibility operations. Finally, both scenarios are compared by computing the improvement ratio of the second scenario (all operations enabled) with the first scenario (using only *selective halting*).

2.1.4 Evaluation metrics

The first metric evaluated is the amount of channels which require drainage of packets. We consider that a channel is required to be drained of packets if it cannot satisfy the upgrade condition due to some incoming target/s. Thus, it has to request incoming dependencies removal to upstream channels for one or more targets going through this channel following the $TCDG_{R_P}$.

This metric considers a channel as *requiring* drainage as soon as a single target brought by the old routing function is not provided any output choice by the new routing function. This results in a conservative metric because it does not consider any particular distribution of packets in the network. In a realistic scenario, depending on the assignment of flits or packets to buffers, channel drainage may not be necessary regardless of the output choices provided by the new routing function. For example, neighbor traffic will not distribute packets towards destinations comprising multiple hops. Hence, regardless of the provided paths by the routing function between source-destination pairs, some of them will never be used by packets towards some destinations.

Another useful metric is the amount of halted flows (i.e. source-destination pairs) for which *selective halting* has been applied during the reconfiguration for some period at source channels. As with the previous metric, this considers a worst case scenario because depending on the traffic pattern, some source-destination pairs may never establish communication. In a real system, this would depend on the job scheduling and task mapping techniques. Notice that injection *selective halting* is applied at some source-destination pairs if and only if no available routes exist between those processing nodes.

2.2 Drained channels and halted flows

The Upstream Progressive Reconfiguration algorithm has been evaluated triggering the RP for each combination of routing algorithms. Figure 2.1 shows the ratio of channels which required drainage of at least one incoming target with respect to the total amount of channels available in a 5×5 mesh network. Each bar represents the final routing algorithm while the initial routing algorithm is in the horizontal axis.

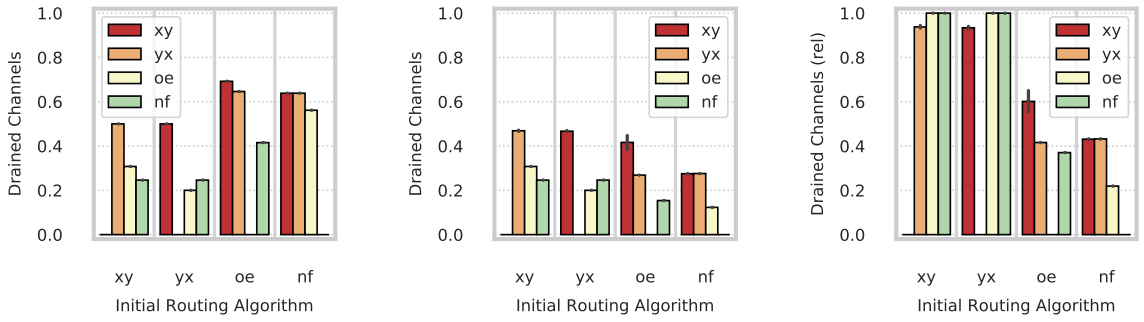
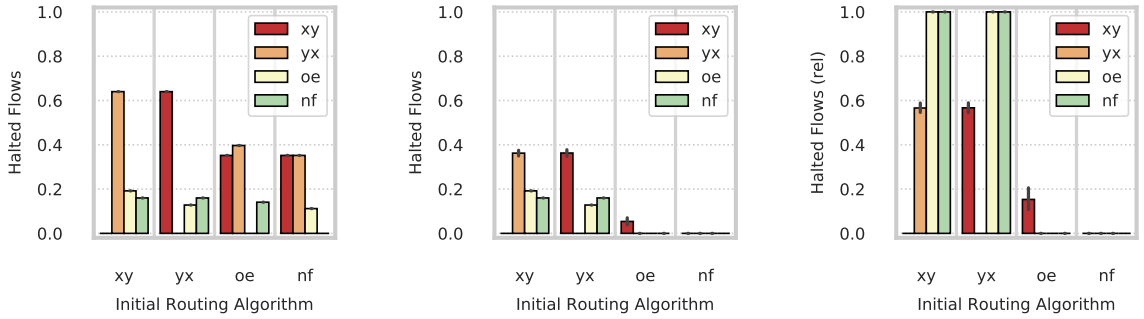


Figure 2.1: Drained channels ratio: (a) Using only *selective halting* of traffic at source channels; (b) Exploiting conformability and compatibility; (c) Exploiting conformability/compatibility with respect to using *selective halting* only.

Figure 2.1a shows the channel ratio requiring drainage if relying solely in *selective halting*. First, the figure indicates that not all channels are required to be drained, this is due to channels yielding a target conforming $Local_{R_I\{c\}}$ with respect to R_P (sink channels always satisfy this). Non-adaptive algorithms *xy* and *yx* have a single route per source-destination pair. Hence, when used as initial routing algorithms,



(a) (b) (c)

Figure 2.2: Halted flows ratio: (a) Using only *selective halting* of traffic at source channels; (b) Exploiting conformability and compatibility; (c) Exploiting conformability/compatibility with respect to using *selective halting* only.

lower channel drainage is observed due to these algorithms restraining the amount of different routes towards the same destination.

On the other hand, using *xy* and *yx* as final routing algorithms increases the drained channels ratio for the same reason. Due to the existence of a single path between each source-destination pair, the ratio of shared dependencies with the initial routing algorithm is lower, specially when combined with partially adaptive routing algorithms *oe* and *nf* as the initial routing algorithms. Adaptiveness degree provided by *oe* and *nf* when used as initial routing algorithms reduce the amount of channels with a target conforming $Local_{R_I\{c\}}$ due to a higher amount of targets going through channels at R_P. Thus, more channels will be drained in these cases (greater than 60%).

Exploiting conformability and compatibility results regarding drained channels are shown in Figure 2.1b. Clearly, this shows great improvements when *oe* and *nf* are used as initial routing algorithms due to the exploitation of conformability to reduce R_P in order to remove unwanted dependencies. In addition, when combined with *xy* and *yx* as final routing algorithms, the amount of drained channels is lowered to less than 45% (lower than 30% for *nf* as initial routing algorithm) due to the great chances of exploiting compatibility within R_I to upgrade channels, increasing the amount of different routes provided by *xy*, *yx*. Greater reduction is obtained when combined with *oe* and *nf* as final routing algorithms (lower than 20%) due to a greater amount of shared dependencies (i.e. portions of paths between source-destination pairs) which increases the amount of channels with target conforming $Local_{R_I\{c\}}$.

Channel drainage ratio when exploiting conformability/compatibility relative to just *selective halting* is shown in Figure 2.1c. It can be observed that greater improvements (i.e. less channels requiring drainage) are obtained combining partially adaptive algorithms *oe*, *nf*. These routing algorithms are maximally extended such that no dependencies may be added to their associated $TCDG$ such that they do not create a cycle. This increases the amount of target conforming $Local_{R_I\{c\}}$ for channels due to a large amount of shared dependencies among both routing functions.

No significant improvement is obtained when algorithms *xy*, *yx* are used as initial routing algorithms with respect to relying only on *selective halting*. This is due to a low amount of target conforming $Local_{R_I\{c\}}$ found among channels. Besides, due to *oe* and nf being maximally extended routing functions, compatibility cannot be exploited to a great extent when these algorithms are used as final routing algorithms. Reduction of these routing algorithms would be desirable in order to leave margin for UPR to add dependencies such that non target conforming $Local_{R_I\{c\}}$ become compatible.

Channel drainage against other reconfiguration schemes can be greatly reduced. For example, OSR always requires all channels to be drained from all packets under R_P for the proposed combinations of initial and final routing algorithms. This is due to the use of the two distinct sets of escape resources provided by R_P and R_I. The former must be used by packets routed under R_P and the later will be used for new injected packets under R_I. In OSR, at any given moment in time, channels may belong to only one of these predefined escape sets. This, in turn, makes OSR unlikely to benefit from the dynamic nature of the reconfiguration process and existing compatibilities between these two escape channel sets.

Figure 2.2 shows halted flows during the reconfiguration process. Figure 2.2a shows halted flows resulting from applying only *selective halting*. Worst cases are combinations between *xy* and *yx* requiring to halt injection of more than 60% of flows due to the lack of alternative routes between each source-destination pair. Hence, a single channel requiring the removal for a particular incoming target triggers the injection *selective halting* of that target at multiple source channels.

Regardless of the initial routing algorithm used, setting *oe* and *nf* as the final routing algorithms, keeps the ratio of halted flows under 20%. This is a direct consequence of the increased amount of routing choices (and portions of paths) which are shared with the initial routing algorithm due to a greater number of paths between each source-destination pair. Thus, a low amount of removal requests is received at upstream channels to remove offending targets towards downstream channels.

When exploiting conformability and compatibility halted flows results are shown in Figure 2.2b. We can observe a great reduction in halted flows required from the combination between *xy* and *yx*, which now is less than 40% which represents a 60% with respect to applying only *selective halting* according to Figure 2.2c. Besides, halted flows for *oe* and *nf* used as the initial routing algorithms are completely avoided in some cases (i.e. 0%) except for combination *oe-xy* which has been brought from around 37% to 8%.

Overall, halted flows ratio is reduced when exploiting conformability/compatibility. However, halted flows reduction is greater for combinations among routing algorithms offering multiple routes between source-destination pairs. This increases the probability that some paths (or portion of paths) followed by packets towards its destination can be shared among both routing algorithms, either directly or by finding a compatible $Local_{R_I\{c\}}$ at channels depending on the extension capability yielded by the final routing algorithm.

Existing reconfiguration schemes do not rely on selective injection halting of flows. Instead, they rely on buffer occupancy backpressure, which may prevent source channels to inject packets towards any destination. This is of special importance because old packets interference with new packets may block new packets forwarding at a given channel. This interference is propagated backwards, resulting in new packets being blocked at source channels.

Proposals such as OSRLite[1] try to alleviate this problem by allowing packets routed under the new routing function to be routed (from a certain point in the network) using the old routing function. However, under a congested scenario this dramatically increases the amount of packets under the old routing function that have to be drained from the escape channel set provided by the new routing function. Thus, delaying RP forward progress significantly.

2.2.1 Additional results

Additionally, we draw some results from exploitation of conformability/compatibility in isolation for each case. Figures 2.3 to 2.10 show the obtained results according to the drained channels and halted flows ratio respectively for each scenario. Finally, figures 2.11 and 2.12 show drained channels and halted flows ratio exploiting all cases of conformability/compatibility.

Tables 2.1 and 2.2 summarize the impact of each sort of exploitation over each combination of initial/final routing algorithms regarding the drained channel and halted flows ratio respectively. We have used the following codes to identify each kind of exploitation:

Table 2.1: Drained channels ratio reduction.

Initial Routing Algorithm	Final Routing Algorithm			
	xy	*yx*	*oe*	*nf*
xy	-	*D*		
yx	*D*	-		
oe	*AD*	*AD*	-	*A*
nf	*AD*	*AD*	*A*	-

Table 2.2: Halted flows ratio reduction.

Initial Routing Algorithm	Final Routing Algorithm			
	xy	*yx*	*oe*	*nf*
xy	-	*D*		
yx	*D*	-		
oe	*ACD*	*AD*	-	*A*
nf	*AD*	*AD*	*A*	-

- A: Exploiting conformability through R''_P to remove dependencies.
- B: Exploiting conformability through R''_I to upgrade channels.
- C: Exploiting compatibility through R'_P to remove dependencies.
- D: Exploiting compatibility through R'_I to upgrade channels.

We can observe that exploiting conformability through R''_I to upgrade channels (i.e. B) does not reduce the drained channels ratio (see Figure 2.5) nor the halted flows (see Figure 2.6) for any combination of routing algorithms tested. However, this sort of exploitation may help the reconfiguration process to complete faster. Similarly, exploiting compatibility through R'_P to remove dependencies has not proven to be effective for the proposed scenarios (see Figures 2.7 and 2.8).

On the other hand, exploiting conformability through R''_P to remove dependencies (i.e. A) has significant effect when using routing algorithms *oe* and *nf* as initial routing algorithms due to a higher amount of paths between each source-destination pair. This allows dependency removal requests to be fulfilled by using an existing alternative route (see Figures 2.3 and 2.4).

Exploiting compatibility through R'_I to upgrade channels (i.e. D) also allows a significant reduction on the drained channel (see Figure 2.9) and halted flows ratio (see Figure 2.10) when using routing algorithms *xy* and *yx* as final routing algorithms. This is due to the low amount of alternative paths provided by these algorithms, allowing the addition of new routing choices in order to provide alternative paths for incoming targets which *xy* and *yx* did not expect originally.

Triggering the reconfiguration process from initial routing algorithms *xy* and *yx* to final routing algorithms *oe* and *nf* is not able to exploit neither conformability nor compatibility. Notice that using only *selective halting*, already lowers the drained channels ratio under 35% and the halted flows ratio under 20% for these combinations. This means that a lot of the routing choices provided by the initial routing algorithms *xy* and *yx* were also provided by final routing algorithms *oe* and *nf* without the need to perform any modifications.

Besides, it has been showed that exploiting conformability through R''_P to remove dependencies (A) and exploiting compatibility through R'_I to upgrade channels (D) are effective in most cases. Nevertheless, initial routing algorithms *xy* and *yx* are maximally reduced and they cannot take any benefit from A. Also, final routing algorithms *oe* and *nf* are maximally extended, as a result, no dependencies can be added through D.

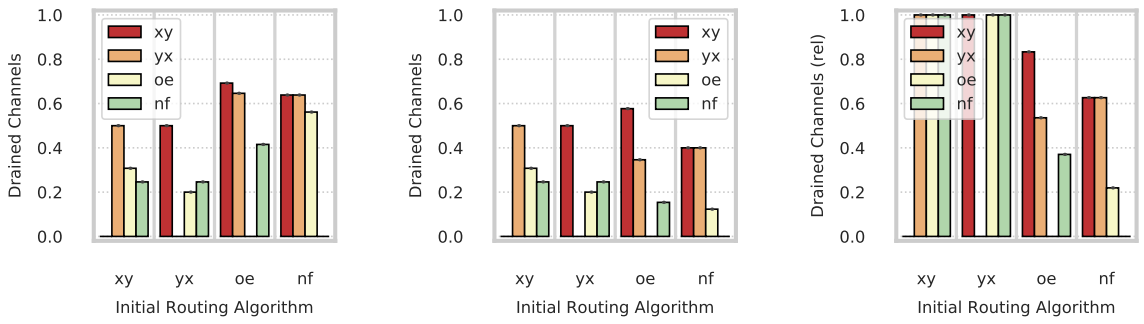
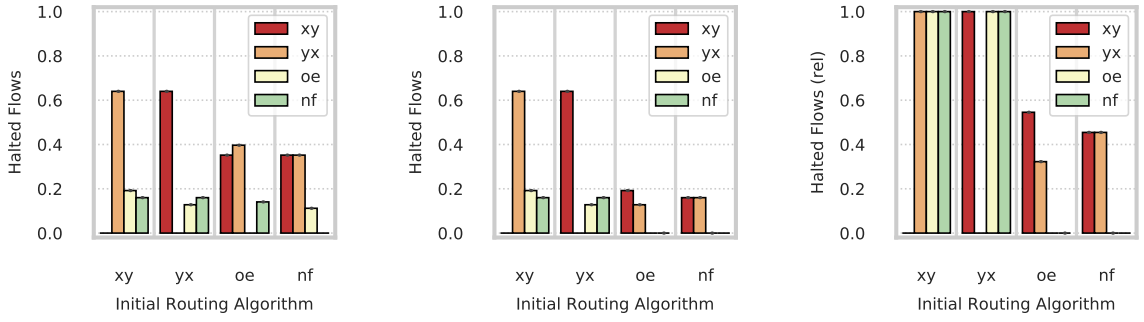
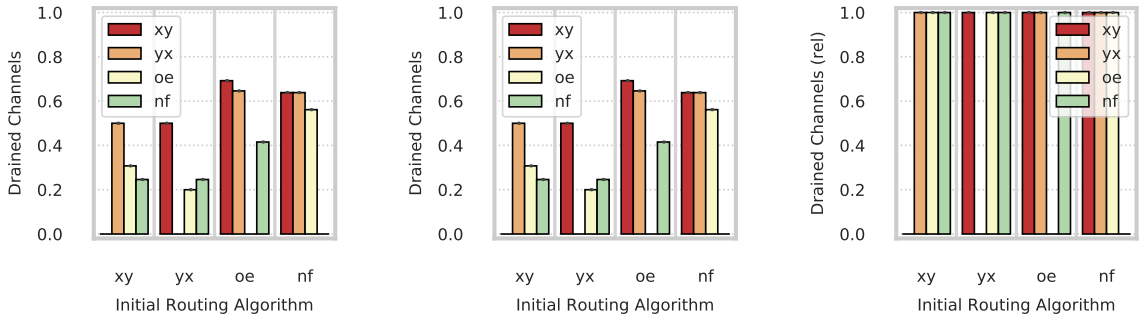


Figure 2.3: Drained channels ratio: (a) Using only *selective halting* of traffic at source channels; (b) Exploiting conformability through R''_P to remove dependencies; (c) Exploiting conformability with respect to using *selective halting* only.



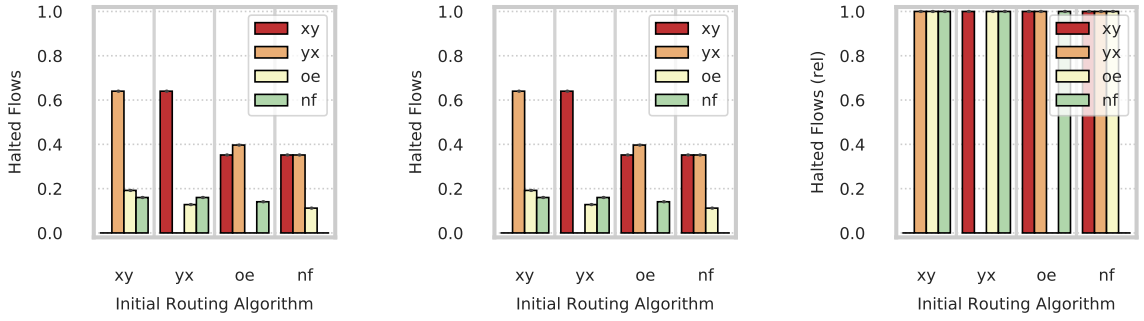
(a) (b) (c)

Figure 2.4: Halted flows ratio: (a) Using only *selective halting* of traffic at source channels; (b) Exploiting conformability through R''_P to remove dependencies; (c) Exploiting conformability with respect to using *selective halting* only.



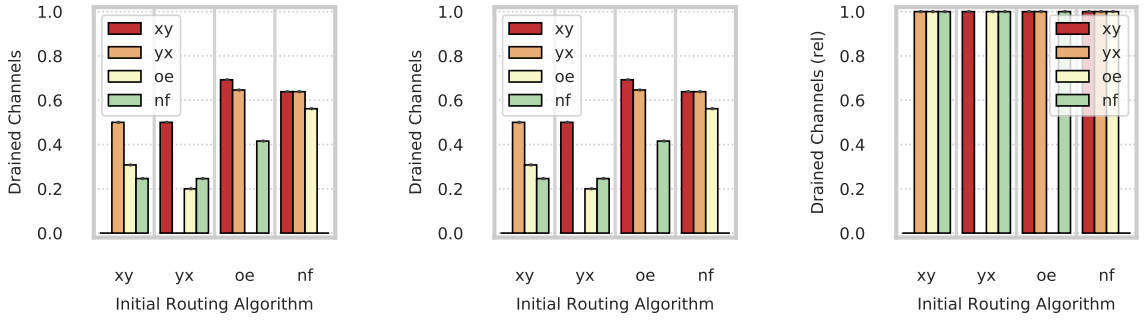
(a) (b) (c)

Figure 2.5: Drained channels ratio: (a) Using only *selective halting* of traffic at source channels; (b) Exploiting conformability through R''_I to upgrade channels; (c) Exploiting conformability with respect to using *selective halting* only.



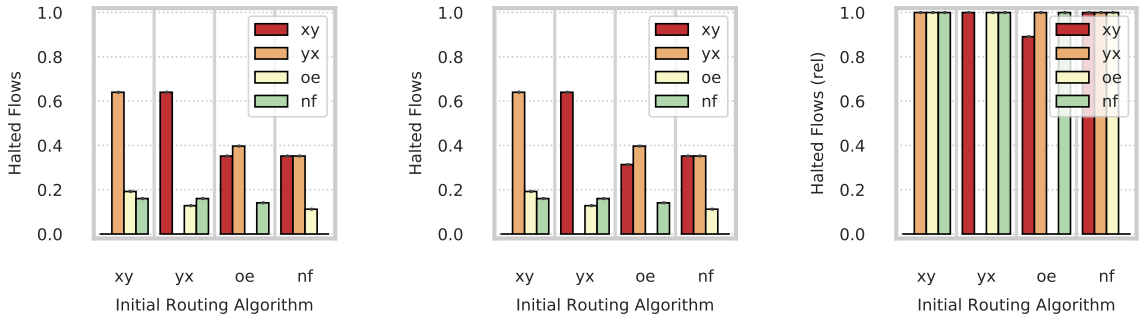
(a) (b) (c)

Figure 2.6: Halted flows ratio: (a) Using only *selective halting* of traffic at source channels; (b) Exploiting conformability through R''_I to upgrade channels; (c) Exploiting conformability with respect to using *selective halting* only.



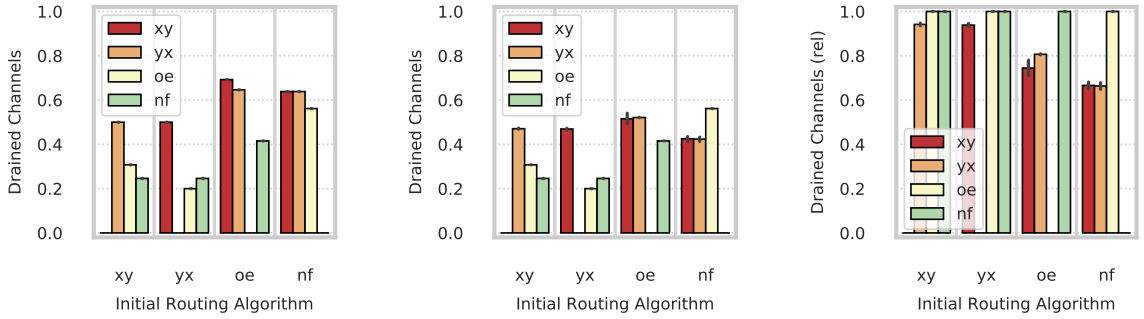
(a) (b) (c)

Figure 2.7: Drained channels ratio: (a) Using only *selective halting* of traffic at source channels; (b) Exploiting compatibility through R'_P to remove dependencies; (c) Exploiting compatibility with respect to using *selective halting* only.



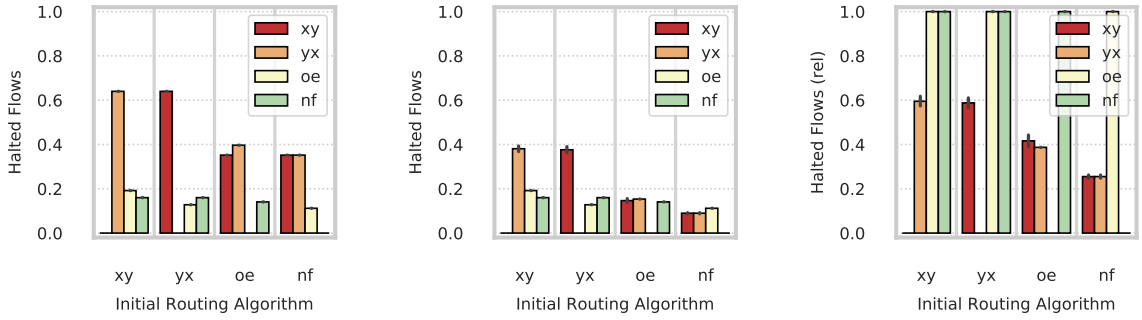
(a) (b) (c)

Figure 2.8: Halted flows ratio: (a) Using only *selective halting* of traffic at source channels; (b) Exploiting compatibility through R'_P to remove dependencies; (c) Exploiting compatibility with respect to using *selective halting* only.



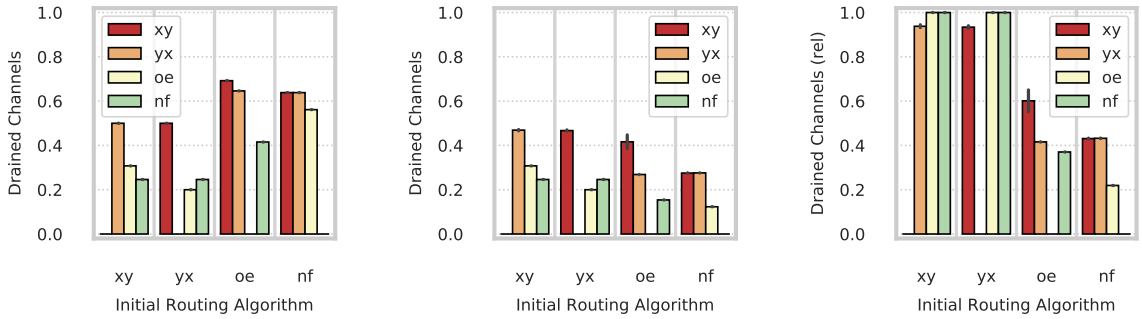
(a) (b) (c)

Figure 2.9: Drained channels ratio: (a) Using only *selective halting* of traffic at source channels; (b) Exploiting compatibility through R'_I to upgrade channels; (c) Exploiting compatibility with respect to using *selective halting* only.



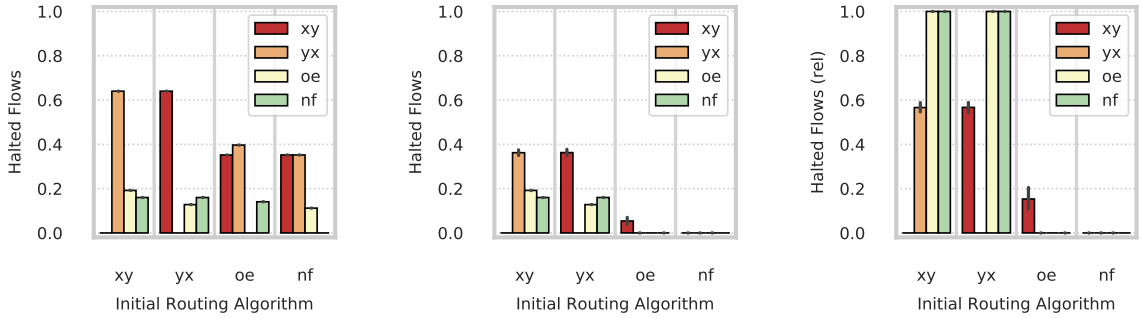
(a) (b) (c)

Figure 2.10: Halted flows ratio: (a) Using only *selective halting* of traffic at source channels; (b) Exploiting compatibility through R'_I to upgrade channels; (c) Exploiting compatibility with respect to using *selective halting* only.



(a) (b) (c)

Figure 2.11: Drained channels ratio: (a) Using only *selective halting* of traffic at source channels; (b) Exploiting conformability and compatibility; (c) Exploiting conformability/compatibility with respect to using *selective halting* only.



(a) (b) (c)

Figure 2.12: Halted flows ratio: (a) Using only *selective halting* of traffic at source channels; (b) Exploiting conformability and compatibility; (c) Exploiting conformability/compatibility with respect to using *selective halting* only.

Bibliography

[1] Marco Balboni, Francisco Trivino, Jose Flich, and Davide Bertozzi. Optimizing the overhead for network-on-chip routing reconfiguration in parallel multi-core platforms. In *2013 International Symposium on System on Chip (SoC)*, pages 1–6. IEEE, 2013.

[2] Ge-Ming Chiu. The odd-even turn model for adaptive routing. *IEEE Transactions on parallel and distributed systems*, 11(7):729–738, 2000.

[3] Christopher J Glass and Lionel M Ni. The turn model for adaptive routing. *ACM SIGARCH Computer Architecture News*, 20(2):278–287, 1992.