2025–2026

L3 INFORMATIQUE

Simulateur de Réseau

Réseaux — Le Routage

Rapport du TP4

CONTRIBUTEURS DU PROJET

Killian Reine Gr A

Mise à jour : 25 février 2026

1 Rappel du sujet

Ce TP porte sur l'implantation d'un programme Java permettant de simuler un réseau de commutateurs et de calculer le chemin le plus court dans une topologie réseau donnée. Le programme doit répondre aux exigences suivantes :

1. Permettre la **saisie graphique du réseau** (commutateurs, liens pondérés, machines terminales).
2. Permettre à l'utilisateur de **sélectionner deux machines** et de calculer l'ensemble des nœuds intermédiaires ainsi que les interfaces utilisées, via l'algorithme de Dijkstra.
3. Afficher le résultat sous forme **graphique et textuelle**.
4. Établir les **tables de routage** de chaque commutateur et les **tables de circuits virtuels** entre deux machines.

L'axe technique principal est la mise en œuvre d'une architecture **hybride C/Java** : un moteur de calcul bas niveau en C, piloté par une interface graphique riche développée en Java Swing.

2 Analyse du sujet

2.1 Décomposition du problème

Le sujet peut être décomposé en quatre grandes problématiques indépendantes mais reliées :

Modélisation du réseau

Un réseau est un graphe pondéré non orienté. Les nœuds sont des commutateurs (*switches*) dotés d'un nombre fini de ports. Les arêtes sont des liens pondérés entre commutateurs. Les machines terminales sont attachées aux commutateurs via un port.

Représentation graphique interactive

L'utilisateur doit pouvoir construire le réseau visuellement : ajouter/supprimer des commutateurs et des liens, positionner les éléments par glisser-déposer, simuler des pannes. La représentation doit être fidèle à la topologie réelle avec affichage des numéros d'interfaces.

Calcul de plus court chemin

L'algorithme de Dijkstra doit être appliqué entre deux commutateurs quelconques. Le résultat doit être visualisé (animation du chemin sur le graphe) et la distance totale affichée. Les tables de routage doivent être calculées pour chaque commutateur.

Circuits virtuels

Pour une liste de liaisons à établir en ordre, il faut déterminer pour chaque commutateur traversé le port d'entrée, le port de sortie, et le VCI (Virtual Circuit Identifier) correspondant. Deux circuits distincts empruntant le même chemin doivent avoir des VCI différents.

2.2 Choix d'architecture

La contrainte de performance pour les calculs de graphe (Dijkstra, tables de routage) nous a orientés vers une **architecture hybride** : le cœur algorithmique est implanté en C (vitesse, gestion mémoire fine), tandis que l'interface graphique et la logique de présentation sont développées en Java Swing. La communication entre les deux couches se fait via un **protocole texte** sur les flux stdin/stdout d'un processus fils.

3 Choix techniques effectués

3.1 Architecture générale : patron MVC

Le projet suit une architecture **Modèle-Vue-Contrôleur** stricte, organisée en packages Java distincts :

— `ui.model` : structures de données pures (réseau, commutateurs, liens, machines, circuits virtuels).

— `ui.view` : tous les composants graphiques (canvas, panneaux, dialogues, renderers).

— `ui.controller` : logique de traitement des événements utilisateur.

— `ui.bridge` : pont Java/C (`CBridge`).

— `ui.palette` : constantes visuelles partagées (couleurs, fontes, dimensions).

Le package `ui.view.renderer` est lui-même subdivisé en sous-packages spécialisés : `renderer.node` pour le rendu des commutateurs et `renderer.machine` pour le rendu des machines terminales. Cette granularité permet de modifier le rendu d'un élément visuel sans toucher aux autres.

3.2 Le moteur C

Structures de données

Le moteur C repose sur deux structures principales définies dans les en-têtes :

— `Switch` : commutateur avec identifiant, tableau de voisins (`Pair*`), état actif/en panne, et champs Dijkstra (`pi_`, `parent_`).

— `Machine` : terminal identifié, pointeur vers son commutateur de rattachement.

— `Pair` : association {commutateur voisin, poids du lien}.

Algorithme de Dijkstra en C

L'implémentation dans `dijkstra.c` suit l'algorithme classique avec tableau de visités. La complexité est en $O(n^2)$ (adapté aux tailles de réseau des TP). Les fonctions principales sont :

— `dijkstra()` : calcule la distance minimale et remplit les champs `pi_` et `parent_`.

— `reconstruct_path()` : reconstruit le chemin depuis les champs `parent_`, retourne un tableau d'identifiants alloué dynamiquement.

— `reset_dijkstra()` : réinitialise les champs avant un nouveau calcul.

Table de routage

Pour chaque commutateur `S`, la table de routage est calculée en lançant `dijkstra()` depuis `S` vers chaque autre commutateur du réseau. Pour chaque destination, on remonte la chaîne `parent_` d'un seul cran pour identifier le **voisin direct** à emprunter et la **distance** correspondante.

Protocole de communication C/Java

Le processus C tourne en boucle et lit des commandes sur `stdin`. Il répond sur `stdout` par `OK` ou `ERR`. Les commandes implémentées sont :

Commande	Effet
`ADD_SWITCH <id> <ports>`	Ajoute un commutateur
`REMOVE_SWITCH <id>`	Supprime un commutateur et ses liens
`ADD_LINK <id1> <id2> <w>`	Ajoute un lien pondéré
`REMOVE_LINK <id1> <id2>`	Supprime un lien
`ADD_MACHINE <mId> <swId>`	Attache une machine à un commutateur
`DIJKSTRA <src> <dst>`	Calcule le plus court chemin
`ROUTING_TABLE <swId>`	Calcule la table de routage
`SET_ACTIVE <id> <0|1>`	Active/désactive un commutateur
`QUIT`	Termine le processus

3.3 Le pont Java/C : CBridge

La classe `CBridge` encapsule toute la communication avec le processus C. Elle maintient un **cache local** du réseau (objet `Network`) synchronisé avec chaque opération pour permettre le rendu graphique sans re-interroger le processus C à chaque frame. Les méthodes Java sont des wrappers directs sur les commandes texte du protocole.

3.4 Modèle Java

La classe `Link`

Chaque lien stocke ses **numéros d'interface** côté A (`portA`) et côté B (`portB`), attribués automatiquement à la création via `nextPort(sw)` qui incrémente depuis le maximum des ports déjà utilisés sur ce commutateur.

La classe `Machine`

Chaque machine stocke son **numéro d'interface** (`port`) sur le commutateur auquel elle est rattachée. Ce port est attribué en continuité avec les ports déjà occupés par les liens, assurant un espace de numérotation global cohérent par commutateur.

La classe `VirtualCircuit`

Modèle léger avec un identifiant auto-incrémenté, les identifiants source et destination, et un état ouvert/fermé.

3.5 Interface graphique

Réseau de démonstration au démarrage

Au lancement, l'application charge automatiquement un **réseau de démonstration** composé de 10 commutateurs et 18 liens via la méthode `loadDemo()`. Ce réseau permet de prendre en main l'interface immédiatement sans construction manuelle.

Barre de menus

La barre de menus en haut de la fenêtre est organisée en trois menus. **Fichier** donne accès à l'effacement complet du réseau et à la fermeture de l'application. **Outils** propose des raccourcis pour basculer rapidement entre les modes Ajouter switch, Ajouter lien et Sélection. **Circuits Virtuels** ouvre le gestionnaire de circuits virtuels.

Rendu du graphe

Le canvas délègue l'affichage à des sous-renderers spécialisés, organisés en deux niveaux hiérarchiques :

- `GridRenderer` : grille de fond.
- `LinkRenderer` : tracé des arêtes, `LinkWeightBadge` pour le badge de poids central, `LinkPortBadge` pour les badges de numéro d'interface près de chaque commutateur.
- `NodeRenderer` : orchestre trois sous-renderers — `NodeShapeRenderer` (ring extérieur, remplissage, bordure selon l'état actif/en panne), `NodeLabelRenderer` (nom du commutateur), `NodeGlowRenderer` (halo de sélection).
- `MachineRenderer` : orchestre trois sous-renderers — `MachineLayoutCalc` (calcul de l'angle de départ optimal), `MachineDotRenderer` (cercle, trait de liaison, label), `MachinePortBadge` (badge du numéro d'interface côté commutateur).

Placement intelligent des machines

La classe `MachineLayoutCalc` implémente un algorithme de **maximisation d'angle** : elle teste 72 angles candidats régulièrement espacés et retient celui qui maximise la distance minimale aux liens existants du commutateur. Cela évite la superposition visuelle entre machines et liens, quel que soit le degré du commutateur.

Table de routage

La fenêtre `RoutingTableDialog` utilise un `JEditorPane` avec rendu HTML/CSS, ce qui permet un affichage propre et scrollable sans recourir à un `JTable` complexe.

Circuits virtuels

La fenêtre `VirtualCircuitDialog` adopte la même approche HTML/CSS. Le tableau sur 3 niveaux de header (Switch $\rightarrow$ IN/OUT $\rightarrow$ PORT/VCI) est généré dynamiquement en HTML avec `colspan` et `rowspan`, ce qui garantit un rendu correct quelle que soit la largeur des colonnes ou le scroll horizontal.

4 Résultats et tests

4.1 Construction et manipulation du réseau

L'interface permet de créer un réseau de commutateurs et de machines en quelques clics. Le glisser-déposer fonctionne correctement, les liens se repositionnent avec les commutateurs. Les numéros d'interface s'affichent correctement sur le graphe : numéros de port sur les arêtes (près du commutateur) et sur les machines (badge sur le cercle).

4.2 Calcul Dijkstra

Le calcul est effectif entre deux machines quelconques. L'animation révèle progressivement les arêtes du chemin en vert. La simulation de panne d'un commutateur est prise en compte : Dijkstra contourne les nœuds inactifs ou signale l'absence de chemin.

Test de cohérence

Sur un réseau à 4 commutateurs (S1, S2, S3, S4) avec les poids représentés dans la topologie du cours, le chemin calculé correspond au chemin optimal attendu.

4.3 Table de routage

La table de routage s'affiche correctement pour chaque commutateur. La meilleure route est mise en évidence en vert. Les entrées passant par des commutateurs en panne sont automatiquement exclues.

4.4 Circuits virtuels

Pour un réseau M1-S1-S2-S4-M4 :

Circuit	**S1**		**S2**		**S4**	
	IN PORT	VCI	IN PORT	VCI	IN PORT	VCI
M1 $\rightarrow$ M4	p(M1)	1	p(S1$\rightarrow$S2)	1	p(S2$\rightarrow$S4)	1
M1 $\rightarrow$ M4	p(M1)	2	p(S1$\rightarrow$S2)	2	p(S2$\rightarrow$S4)	2

Les deux circuits empruntant le même chemin ont les mêmes ports IN/OUT mais des VCI distincts (1 et 2), ce qui est conforme à la sémantique des circuits virtuels.

5 Conclusion

5.1 Bilan

Ce projet a permis de mettre en œuvre un simulateur de réseau complet répondant à toutes les exigences du sujet. L'architecture hybride C/Java s'est avérée pertinente : elle sépare clairement les responsabilités entre la performance des calculs et la richesse de l'interface. Le patron MVC appliqué côté Java rend le code modulaire et facilement extensible.

Les fonctionnalités implémentées couvrent l'intégralité des points obligatoires (saisie graphique, Dijkstra, tables de routage) et les points facultatifs (machines terminales, interfaces réseau, circuits virtuels). Le refactoring des renderers en sous-classes spécialisées renforce la maintenabilité et facilite toute évolution future de l'apparence.

5.2 Difficultés rencontrées

Communication C/Java

La synchronisation entre le processus C et la vue Java a nécessité de concevoir un protocole de communication robuste et de maintenir un cache Java cohérent. Les cas d'erreur (processus C qui ne répond pas, commandes invalides) ont demandé une gestion attentive.

Numérotation des interfaces

La gestion d'un espace de ports partagé entre liens et machines sur un même commutateur a été délicate à mettre en place, en particulier pour garantir la cohérence lors de suppressions et de rechargements depuis un fichier.

Header du tableau de circuits virtuels

La tentative de réaliser un header groupé sur 3 niveaux avec un `JTable` Swing s'est heurtée à des problèmes de synchronisation du scroll et d'affichage. La solution retenue (rendu HTML via `JEditorPane`) a simplifié considérablement l'implémentation tout en donnant un rendu plus propre.

Placement des machines

Le positionnement automatique des machines autour des commutateurs fortement connectés posait des problèmes de lisibilité. La classe `MachineLayoutCalc` résout ce problème en cherchant l'angle de départ qui maximise l'écart aux liens existants.

ANNEXES

MODE D'EMPLOI DE L'INTERFACE

Guide complet d'utilisation du simulateur
de réseau avec routage et circuits virtuels.

Annexe A — Mode d'emploi de l'interface graphique

Cette annexe constitue le guide d'utilisation complet du simulateur. Elle décrit chaque fonctionnalité accessible depuis l'interface graphique, dans l'ordre naturel d'utilisation.

A.1 — Vue générale de l'application

L'interface principale est divisée en deux zones :

— **Le panneau latéral gauche** (*SidePanel*) : regroupe tous les outils de construction du réseau, le moteur de calcul de chemin Dijkstra, et les informations sur le réseau courant.

— **Le canvas central** (*GraphCanvas*) : surface de dessin interactive sur laquelle le réseau est représenté graphiquement. On y place les commutateurs, on y trace les liens, et on y visualise les chemins calculés.

La **barre de menus** en haut de la fenêtre est organisée en trois menus :

— **Fichier** : effacement complet du réseau (*Vider tout*) et fermeture de l'application (*Quitter*).

— **Outils** : raccourcis pour basculer entre les modes *Ajouter des Switch*, *Ajouter des Lien* et *Mode Sélection*, en complément des boutons du panneau latéral.

— **Circuits Virtuels** : ouvre le gestionnaire de circuits virtuels.

Au démarrage, un **réseau de démonstration** composé de 10 commutateurs et 18 liens est automatiquement chargé afin de permettre une prise en main immédiate.

A.2 — Construction du réseau

Ajouter un commutateur

1. Dans le panneau latéral, sélectionner le mode **Ajouter un switch** (bouton *+ Switch*).
2. Choisir le nombre de ports souhaité via le sélecteur numérique (de 1 à 16 ports).
3. Cliquer à l'emplacement voulu sur le canvas : un nouveau commutateur apparaît, nommé automatiquement `SwN` où `N` est son identifiant.

Il est aussi possible d'effectuer un **clic droit** sur une zone vide du canvas et de choisir *Ajouter switch ici* dans le menu contextuel.

Ajouter un lien entre deux commutateurs

1. Sélectionner le mode **Lier** (bouton *Lier*).
2. Cliquer sur le commutateur source : il se surligne.
3. Cliquer sur le commutateur destination : une boîte de dialogue demande le **poids du lien** (entier entre 1 et 9999).
4. Confirmer : le lien est tracé entre les deux commutateurs, le poids est affiché au milieu de l'arrête, et les **numéros d'interface** apparaissent près de chaque commutateur.

Raccourci : un double-clic sur un commutateur active directement le mode liaison avec ce commutateur comme source.

Ajouter une machine

1. Sélectionner le mode **Machine** (bouton *+ Machine*).
2. Cliquer sur le commutateur auquel rattacher la machine.
3. La machine est ajoutée si le commutateur dispose encore d'un port libre. Son numéro d'interface est affiché sur son cercle, côté commutateur. Le badge `x/y` sur le commutateur se met à jour (machines utilisées / total de ports).

Si tous les ports sont occupés, un message d'avertissement s'affiche. Les machines sont automatiquement placées autour du commutateur en évitant la superposition avec les liens existants.

A.3 — Lecture du graphe affiché

— **Commutateurs** : représentés par des cercles. Un commutateur actif s'affiche normalement ; un commutateur en panne affiche une croix rouge et ses liens sont ignorés dans les calculs.

— **Liens** : lignes entre commutateurs. Le poids est affiché dans un badge au centre. Les **numéros d'interface** (ex. `0`, `1`, `2`...) sont affichés près de chaque extrémité, légèrement décalés perpendiculairement au lien.

— **Machines** : petits cercles satellites reliés à leur commutateur par un trait. Le numéro d'interface est affiché dans un badge sur le cercle machine, côté commutateur, indiquant le port utilisé sur le switch. Les machines sont positionnées automatiquement dans les espaces angulaires libres entre les liens.

— **Chemin Dijkstra** : lorsqu'un chemin est calculé, les liens qui le composent s'illuminent en vert avec une animation progressive. Les badges de port sur ce chemin passent également en vert.

A.4 — Interactions sur le canvas

Déplacer un commutateur

En mode **Sélection**, maintenir le clic gauche enfoncé sur un commutateur et le glisser vers sa nouvelle position. Les liens se repositionnent automatiquement.

Menu contextuel (clic droit)

Un clic droit sur un commutateur affiche un menu avec les options :

— *Ajouter machine* : ajoute immédiatement une machine sur ce commutateur.

— *Simuler panne / Réparer* : bascule l'état actif/en panne du commutateur.

— *Table de routage* : ouvre la fenêtre de table de routage pour ce commutateur.

— *Supprimer switch* : supprime le commutateur et tous ses liens.

Un clic droit sur un lien affiche :

— *Modifier poids* : ouvre une boîte de saisie pour changer le poids du lien.

— *Supprimer lien* : supprime le lien.

Mode suppression

Le mode **Supprimer** (bouton *Supprimer*) permet de supprimer un commutateur ou un lien d'un simple clic gauche.

A.5 — Calcul du plus court chemin (Dijkstra)

La section **Dijkstra** du panneau latéral permet de trouver le chemin le plus court entre deux machines.

1. Sélectionner la **machine source** dans la liste déroulante *De*.
2. Sélectionner la **machine destination** dans la liste déroulante *Vers*.
3. Cliquer sur le bouton **Calculer**.
4. Le résultat s'affiche : distance totale et chemin animé en vert sur le canvas.

Si aucun chemin n'existe (réseau déconnecté ou commutateurs en panne sur le trajet), un message *Aucun chemin !* apparaît.

A.6 — Table de routage

La table de routage d'un commutateur est accessible : via le **clic droit** sur le commutateur → *Table de routage*.
La fenêtre qui s'ouvre présente un tableau HTML avec :

— Une ligne par commutateur destination du réseau.

— Une colonne par voisin direct du commutateur courant.

— La **distance** pour atteindre chaque destination via chaque voisin.

— La **meilleure route** mise en évidence (texte en vert gras).

A.7 — Circuits Virtuels

Le gestionnaire de circuits virtuels est accessible depuis le menu **Circuits Virtuels** → **Gérer les circuits virtuels**.

Structure du tableau

Le tableau affiche une ligne par circuit ouvert, avec pour chaque commutateur traversé :

— Le numéro de port **IN** (entrée sur le commutateur).

— Le **VCI** (Virtual Circuit Identifier) attribué à ce commutateur pour ce circuit.

— Le numéro de port **OUT** (sortie du commutateur).

— Le VCI sortant (identique au VCI entrant sur un même commutateur).

Les commutateurs non traversés par le circuit affichent —. Les circuits fermés apparaissent en italique grisé avec la mention `[F]`.

Ouvrir un circuit virtuel

1. Sélectionner la **machine source** dans le menu déroulant *De*.
2. Sélectionner la **machine destination** dans le menu déroulant *Vers*.
3. Cliquer sur **+ Ouvrir VC**.
4. Le circuit apparaît dans le tableau avec ses ports et VCI calculés automatiquement à partir des numéros d'interface réels du modèle.

Ouvrir/Fermer un circuit

1. Renseigner le numéro du circuit à modifier dans le sélecteur **VC n°**.
2. Cliquer sur **Ouvrir** / **Fermer** : le circuit bascule entre ouvert et fermé.

Supprimer un circuit

1. Renseigner le numéro du circuit dans le sélecteur **VC n°**.
2. Cliquer sur **Supprimer**.

Logique des ports et VCI

Les numéros de port IN et OUT sont déterminés directement depuis le modèle du réseau :

— Pour le commutateur source : IN = port de la machine source, OUT = port du lien vers le prochain commutateur.

— Pour le commutateur destination : IN = port du lien depuis le commutateur précédent, OUT = port de la machine destination.

— Pour un commutateur intermédiaire : IN = port du lien entrant, OUT = port du lien sortant.

Les VCI sont attribués séquentiellement par commutateur : chaque nouveau circuit qui traverse un commutateur reçoit le prochain VCI disponible sur ce commutateur. Deux circuits empruntant le même chemin auront ainsi les mêmes ports IN/OUT mais des VCI différents.

A.8 — Simulation de panne

Il est possible de simuler la panne d'un commutateur :

— **Clic droit** sur le commutateur → *Simuler panne* : le commutateur affiche une croix rouge et est ignoré dans tous les calculs de chemin.

— **Clic droit** sur un commutateur en panne → *Réparer* : le commutateur redevient actif.

Les tables de routage et les circuits virtuels tiennent compte de l'état des commutateurs : aucun chemin passant par un commutateur en panne ne sera calculé.