Relazione finale di Didattica e laboratorio di Calcolatori

Prof. Francesco Vaschetto
Tirocinante Maria Grazia Maffucci
Classe di concorso A042
22 maggio 2013

Scratch e la sincronizzazione di processi

L'utilizzo di Scratch per la simulazione di realtà complesse può risultare un valido aiuto per la comprensione graduale delle stesse. In questo progetto si proporrà di simulare il funzionamento di una rete basata sul protocollo Token ring senza trattare la reale implementazione tecnica del protocollo stesso. Sfruttando le potenzialità di Scratch gli studenti saranno indirizzati ad una comprensione concettuale del funzionamento di questo tipo di rete, consentendo di porre le basi per una successiva trattazione tecnica del problema.

Il progetto può esse utilizzato sia in una classe seconda della scuola superiore di secondo grado, nell'ambito della materia Tecnologie Informatiche, sia durante il triennio, parallelamente alla spiegazione del funzionamento delle reti di computer e dei relativi protocolli.

L'introduzione del caso di studio è volutamente breve per consentire l'utilizzo del progetto anche in una classe iniziale senza necessariamente soffermarsi sui particolari tecnici del funzionamento di una rete Token ring. Se invece dovesse essere usato in una classe del triennio sarebbe opportuna una maggior precisione e completezza della spiegazione del protocollo.

Il caso di studio si presta particolarmente bene ad essere trattato con degli script che richiedono una sincronizzazione tramite messaggi e, per consentire una più semplice comprensione del funzionamento del protocollo, almeno a livello superficiale, sarebbe opportuno integrare la spiegazione con un role game fatto direttamente dagli studenti. L'azione diretta da parte dei discenti per comprendere quali sono le diverse fasi descritte dal protocollo, aiuterà la comprensione dello stesso e farà intuitivamente comprendere la necessità di utilizzare i messaggi per sincronizzare i vari script.

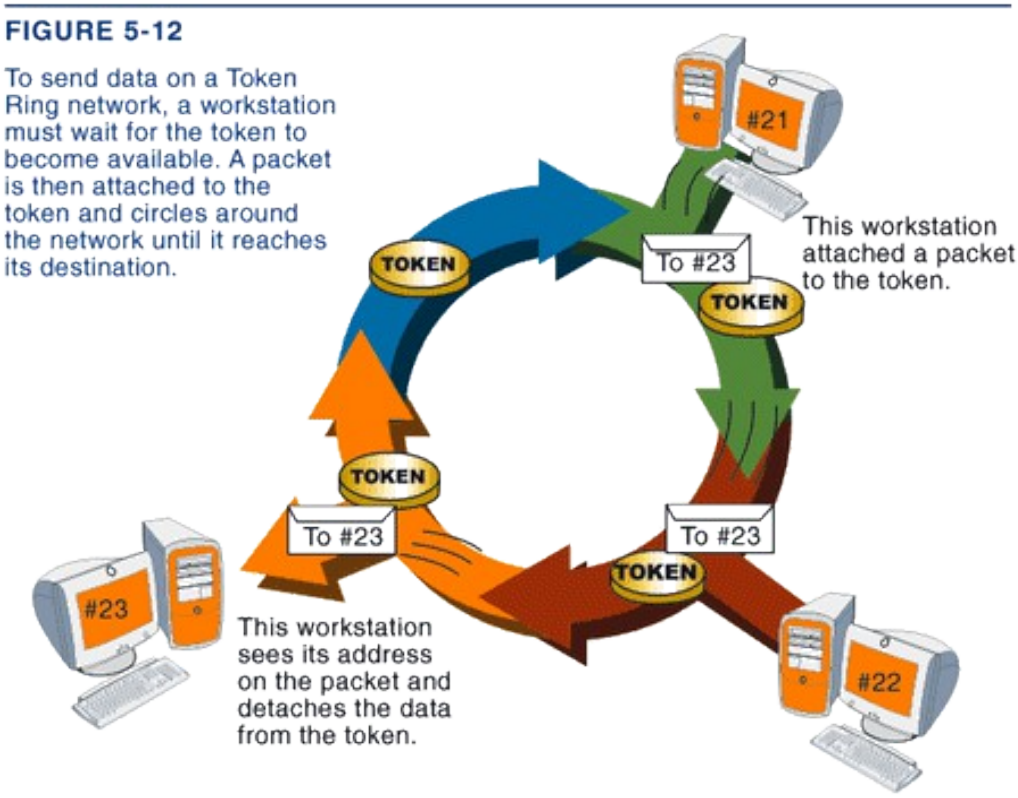
L'utilizzo di Scratch in questo progetto non ha la pretesa di simulare realmente il funzionamento di una rete Token ring, ma consente comunque di giungere ad una comprensione dello stesso per poi eventualmente passare ad una implementazione successiva tramite altri linguaggi di programmazione. Il vantaggio maggiore dell'utilizzo di Scratch rimane comunque la possibilità di evitare gli errori sintattici tipici dei linguaggi di programmazione, consentendo allo studente di concentrarsi completamente sulla logica del problema. Questo tipo di approccio potrebbe rivelarsi estremamente proficuo per sviluppare le capacità logiche degli studenti, anche se necessariamente dovranno essere acquisite successivamente la precisione e la pazienza richieste dai classici linguaggi di programmazione.

Caso di studio

Token ring

Una rete Token ring, che si basa sullo standard IEEE 802.5, è una tipologia di rete che concettualmente possiamo immaginare come un anello, in cui la determinazione del computer che ha il diritto a trasmettere i dati avviene tramite il possesso di un particolare messaggio detto **token**; solo il computer che ha ricevuto questo messaggio può iniziare a trasmettere dei dati nella rete.

Ogni computer è collegato ad altri due formando quello che idealmente possiamo immaginare come un anello.



Quando un computer riceve il token, se ha un messaggio da trasmettere varierà opportunamente il token stesso "attaccandogli" i dati da trasmettere e aggiungendo tutte le informazioni necessarie per raggiungere il computer destinatario collegato alla stessa rete, creando quello che viene chiamato un **frame**.

Affinché il messaggio raggiunga il destinatario, il computer mittente lo invierà al suo "vicino" il quale a sua volta lo spedirà al computer successivo collegato a lui e così via, seguendo l'anello logico di collegamento fino al raggiungimento del destinatario. Quest'ultimo tratterrà i dati e modificherà il messaggio trasformandolo in una conferma di ricezione (**acknowledgment**) destinata al mittente del messaggio. In questo modo il mittente originario, vedendosi confermata la spedizione, rilascerà il token consegnandolo al computer "vicino".

Un qualsiasi nodo collegato all'anello logico che riceverà il token senza avere dati da trasmettere o riceverà un messaggio non indirizzato a lui non farà altro che inoltrarlo al computer vicino.

Sincronizzazione dei processi

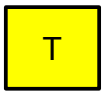
Preparazione dell'ambiente

Questo caso di studio si presta particolarmente per essere trattato mediante la sincronizzazione dei processi temporizzata e/o con scambio di messaggi.

Proveremo ad implementare inizialmente una versione semplificata dove un computer che riceve il token ha sempre un messaggio da trasmettere. In questo modo il token girerà nell'anello e verrà catturato da tutti i computer per effettuare una trasmissione.

La seconda versione invece richiede una gestione più accurata della sincronizzazione dei processi per poter prevedere il caso in cui un computer che riceve il token non abbia nulla da trasmettere, e quindi ceda immediatamente il token al computer vicino.

In entrambe le soluzioni si dovranno prevedere:

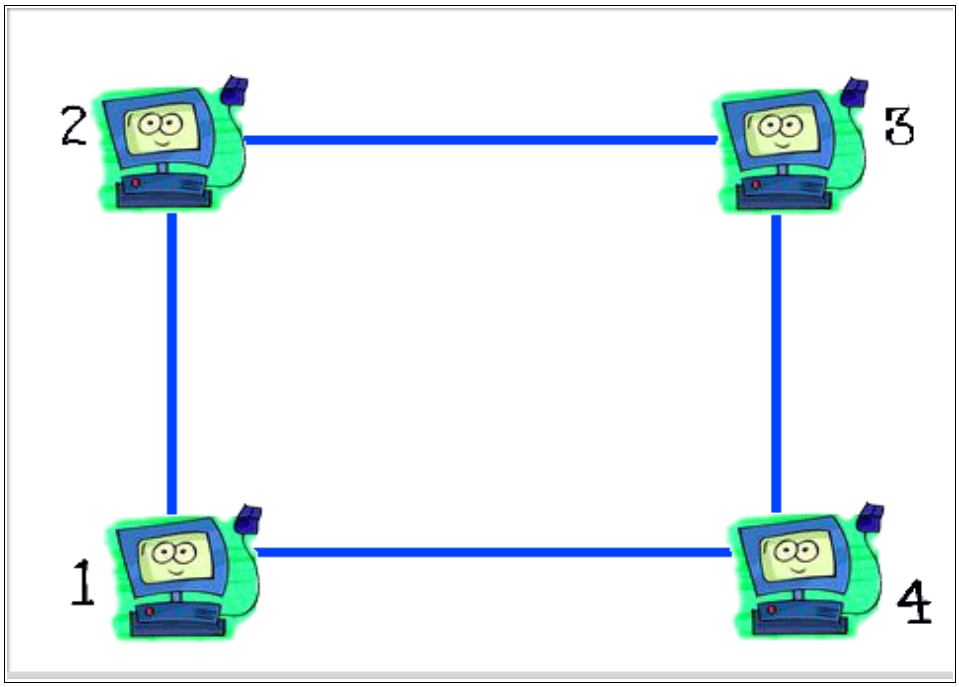
1. tre tipologie di messaggi che potranno essere rappresentati con tre costumi diversi di uno stesso sprite:
   - il token 

   - il messaggio da spedire 

   - la conferma di ricezione 

2. i computer collegati che simuleranno l'anello, eventualmente affiancati da un numero identificativo, ognuno rappresentato con uno sprite diverso:

3. la simulazione di un semplice anello di computer, disegnando opportunamente lo stage e simulando con delle linee il collegamento fra i nodi:



Prima simulazione

Questa prima simulazione prevede che il token venga sequenzialmente passato da un computer al successivo ed ogni computer in possesso del token spedirà un messaggio ad un altro computer scelto a caso fra i tre rimanenti. In questa prima simulazione non viene previsto il caso in cui un computer che riceve il token non abbia un messaggio da spedire.

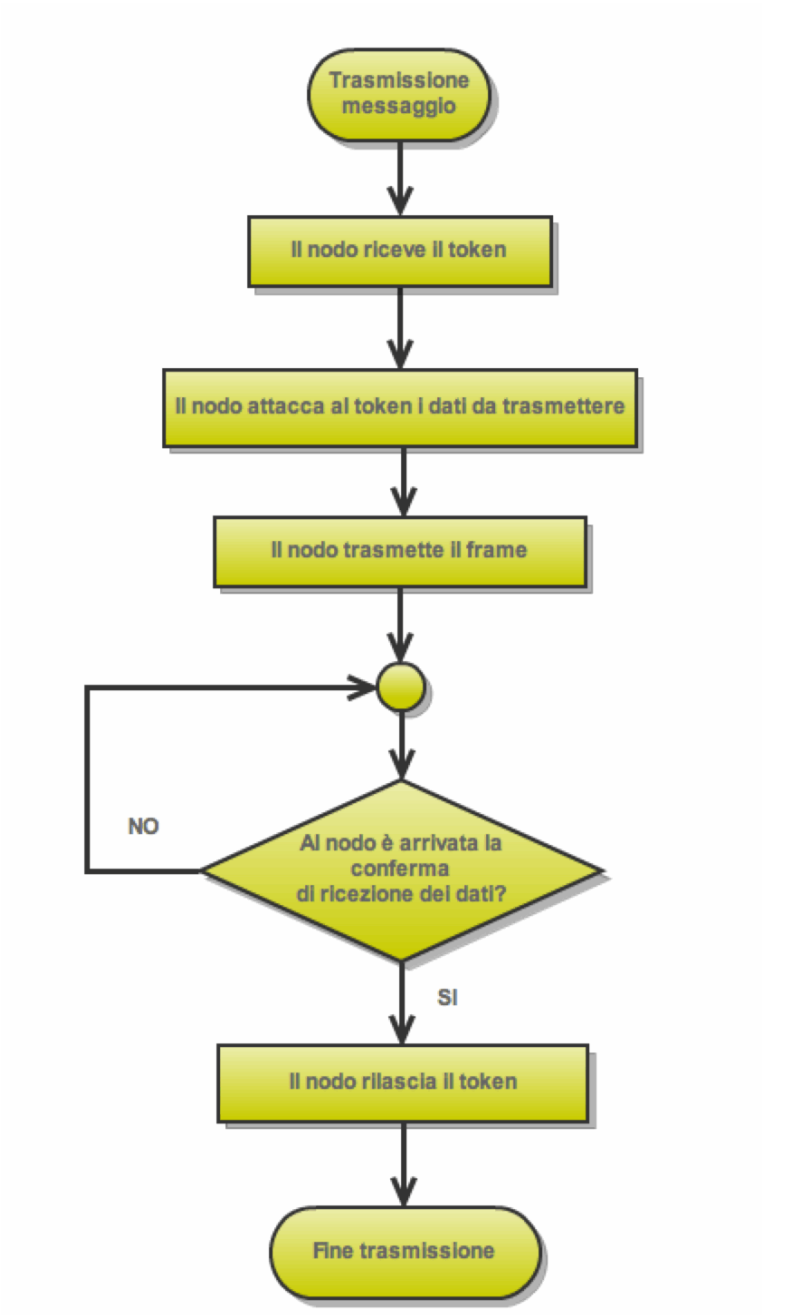
Per simulare il funzionamento della rete Token ring lo sprite utilizzato per rappresentare le tre tipologie di messaggi dovrà cambiare costume in base alla fase in cui si trova la spedizione (invio del token, del messaggio o della conferma), mentre i quattro computer simuleranno la trasmissione o ricezione di uno dei tre tipi di messaggi utilizzando dei fumetti per avvisare la ricezione del token, l'invio di un messaggio o la conferma di un messaggio.

Si potranno quindi individuare due fasi logicamente distinte:

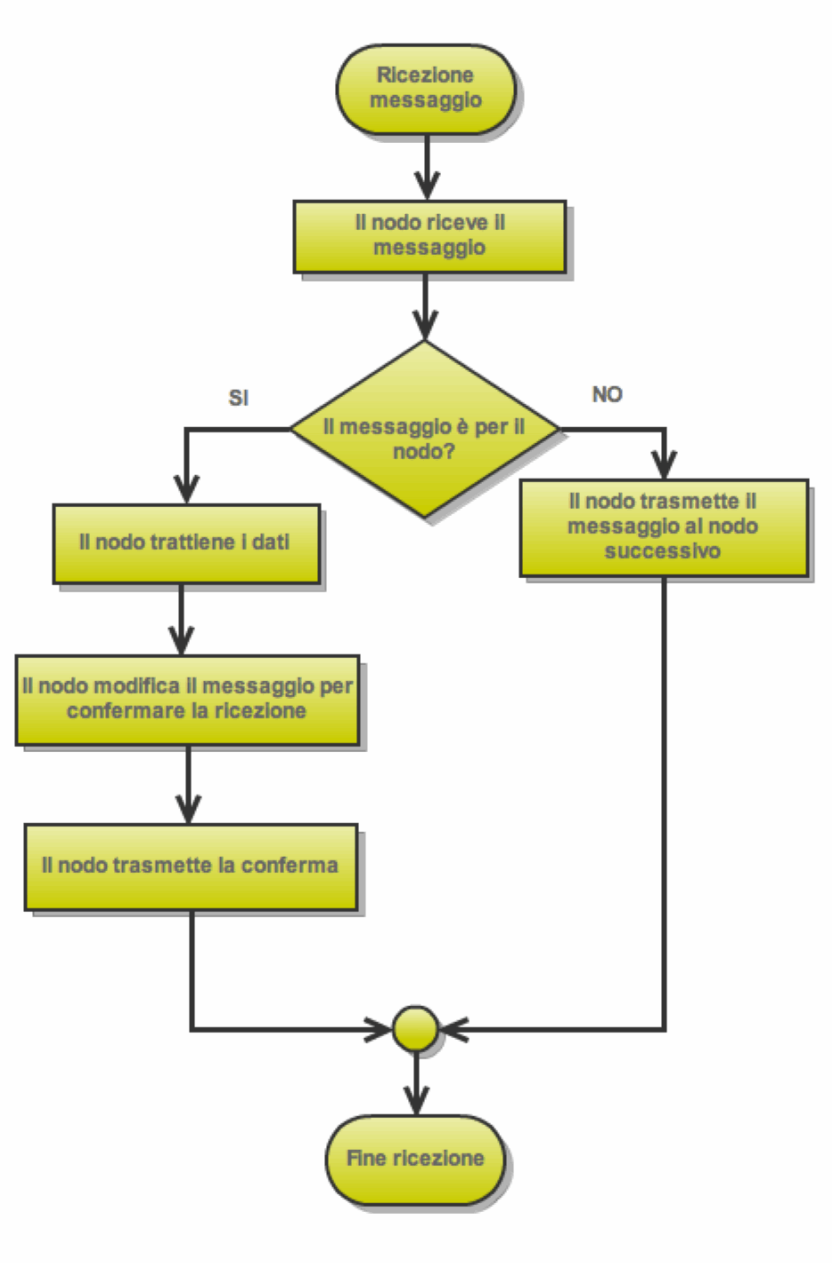
1. una fase di trasmissione in cui un computer potrà trasmettere:
    - un token al computer successivo;
    - un messaggio per un altro computer presente nella rete a seguito del ricevimento del token;
    - una conferma ad un messaggio ricevuto;
2. una fase di ricezione in cui un computer potrà ricevere:
    - un token da un computer che logicamente lo precede;
    - un messaggio a lui destinato;
    - una conferma ad un messaggio da lui spedito precedentemente.

Di seguito sono state schematizzate logicamente, a titolo esemplificativo, la fase di trasmissione di un messaggio e la fase di ricezione di un messaggio.

Nella fase di trasmissione di un messaggio il nodo deve aver ricevuto il token, deve trasformare il token in un messaggio contenente i dati e le informazioni di instradamento e deve rimanere in attesa della conferma di ricezione, giunta la quale potrà rilasciare il token inviandolo al computer successivo.



Nella fase di ricezione di un messaggio il nodo dovrà verificare se era indirizzato a lui e, in questo caso dovrà estrarre i dati, trasformare il messaggio in una conferma di ricezione indirizzata al mittente originario e spedirla inviandola al nodo logicamente successivo. Nel caso in cui il messaggio non fosse indirizzato al nodo ricevente, il messaggio verrà immediatamente inviato al nodo successivo.



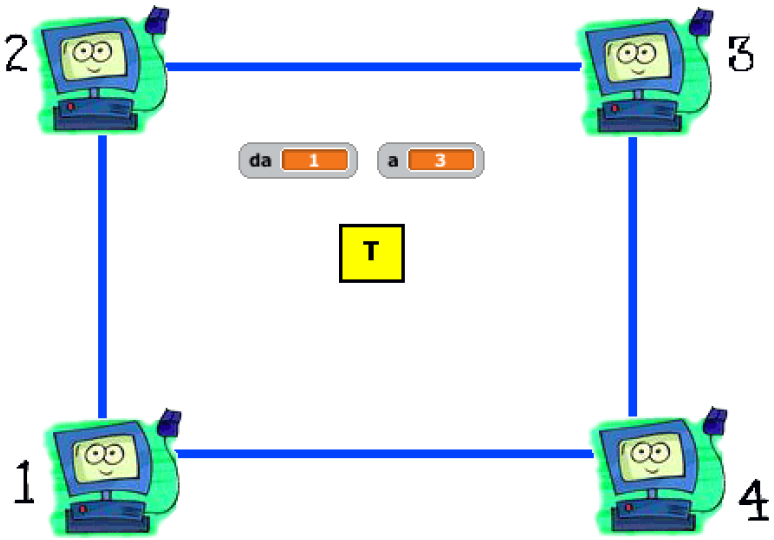
Implementazione della prima simulazione

Viene di seguito fornita una possibile implementazione di questa prima versione.

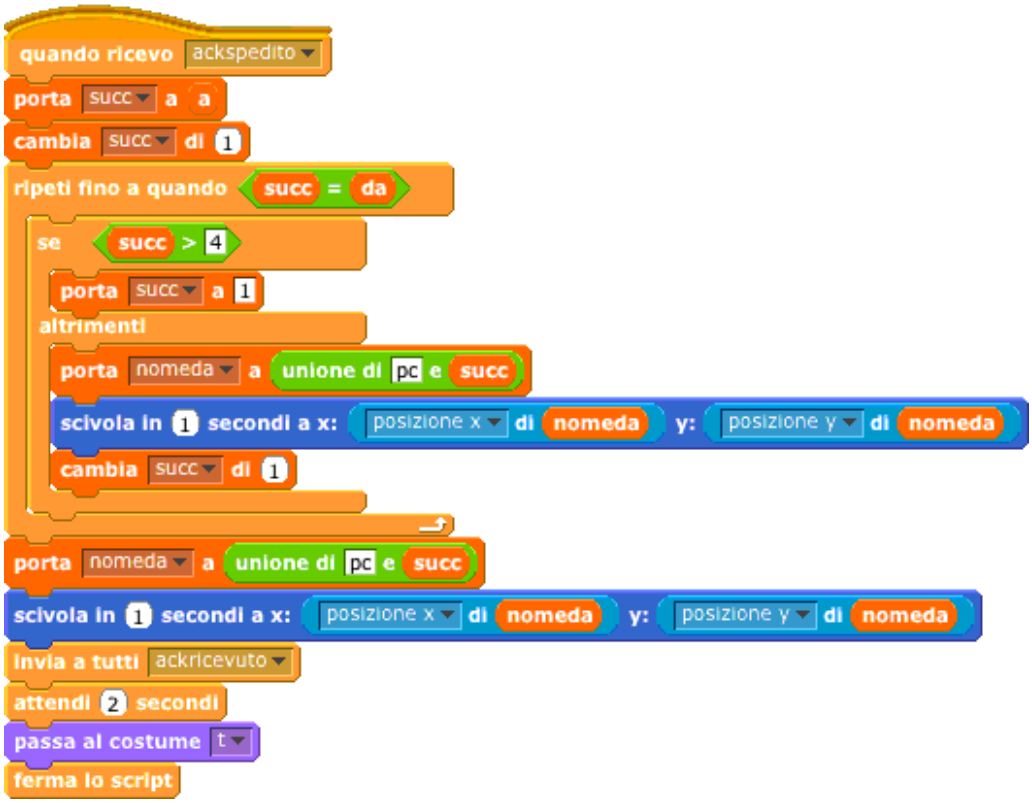
Sono stati previsti:

- uno sprite *token* che assumerà i tre costumi dei diversi messaggi;
- uno sprite per ogni computer (*pc1*, *pc2*, *pc3*, *pc4*);
- uno sprite per i singoli numeri associati ai computer;
- la variabile *da* che conterrà il numero del computer mittente;
- la variabile *a* che conterrà il numero del computer destinatario, selezionato in modo casuale;
- la variabile *succ* che conterrà il numero del computer logicamente successivo durante la fase di movimento dello sprite *token* fra i vari computer;
- la variabile *nomeda* che conterrà il nome dello sprite che identifica il computer mittente;
- la variabile *nomea* che conterrà il nome dello sprite che identifica il computer destinatario;
- il messaggio *token* inviato dallo sprite *token* a tutti gli sprite che identificano un computer per segnalare che uno di loro è in possesso del token; questo messaggio attiverà in ogni sprite computer una verifica per determinare se si è i possessori del token e la relativa visualizzazione di un fumetto che esplicita tale possesso da parte di uno solo dei computer;
- il messaggio *msgricevuto* inviato dallo sprite *token* a tutti gli sprite che identificano un computer per segnalare che uno di loro ha ricevuto un messaggio; questo messaggio attiverà in ogni sprite computer una verifica per determinare se si è i destinatari del messaggio e la relativa visualizzazione di un fumetto che esplicita la situazione;
- il messaggio *ackspedito* inviato dallo sprite *token* a se stesso per gestire gli spostamenti dello sprite associati alla fase di spedizione di una conferma;
- il messaggio *ackricevuto* inviato dallo sprite *token* a tutti gli sprite che identificano un computer per segnalare che uno di loro ha ricevuto una conferma; questo messaggio attiverà in ogni sprite computer una verifica per determinare se si è i destinatari della conferma e la relativa visualizzazione di un fumetto che esplicita la situazione.

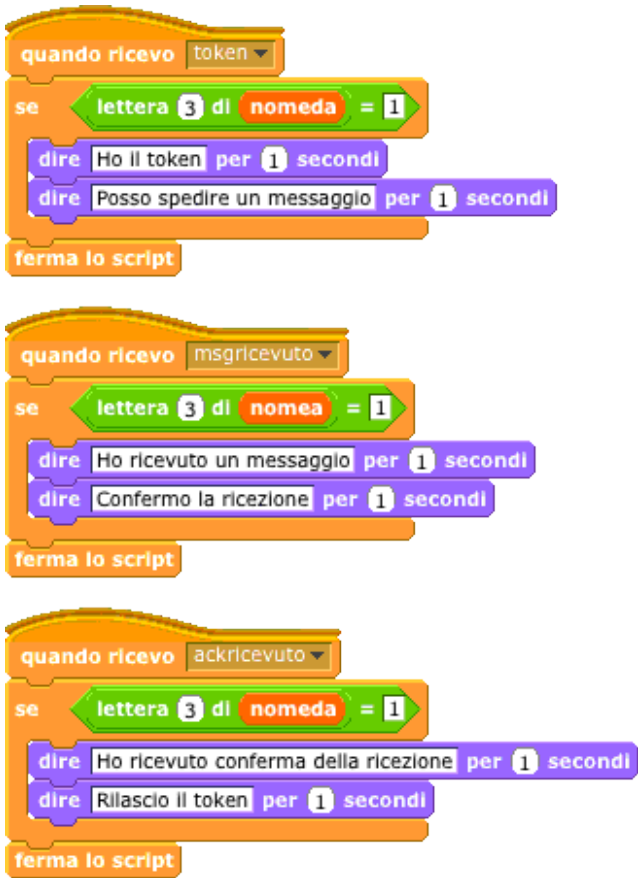
L'ambiente finale sarà il seguente:



Script dello sprite *token*:


quando ricevo ackspedito
porta succ a a
cambia succ di 1
ripeti fino a quando succ = da
se succ > 4
porta succ a 1
altrimenti
porta nomeda a unione di pc e succ
scivola in 1 secondi a x: posizione x di nomeda y: posizione y di nomeda
cambia succ di 1
porta nomeda a unione di pc e succ
scivola in 1 secondi a x: posizione x di nomeda y: posizione y di nomeda
invia a tutti ackricevuto
attendi 2 secondi
passa al costume t
ferma lo script

Script dello sprite *pc1* (gli script degli sprite *pc2*, *pc3*, *pc4* sono analoghi):



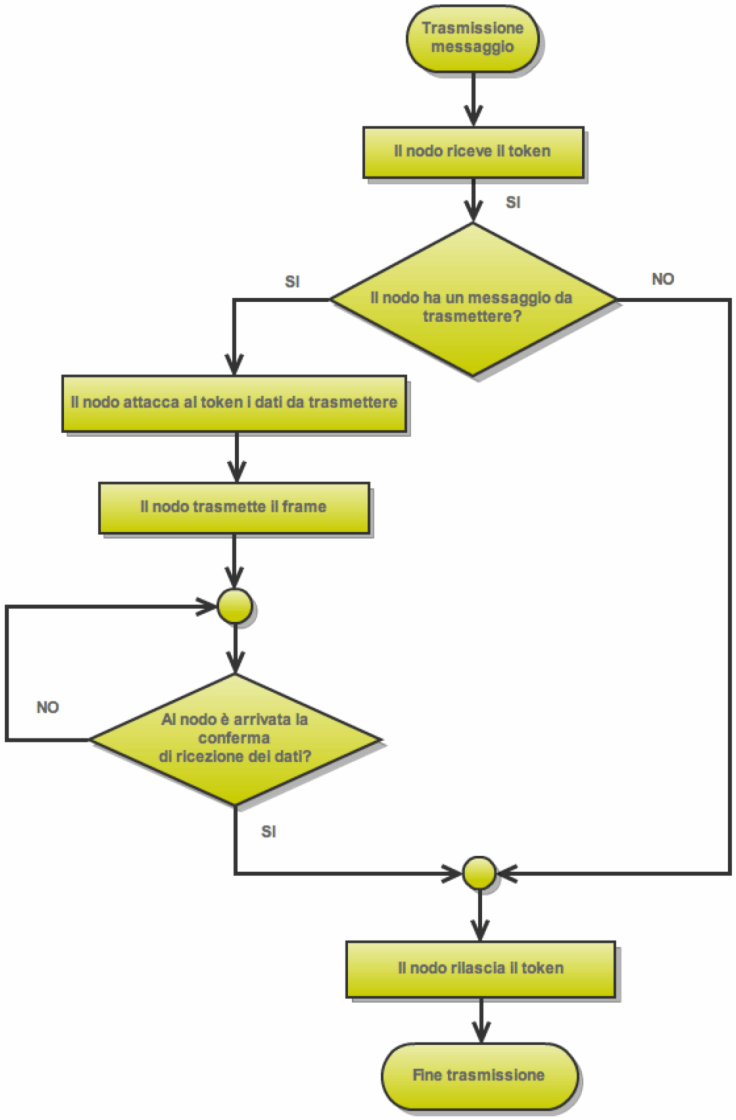
Seconda simulazione

Questa seconda simulazione prevede che il token venga sequenzialmente passato da un computer al successivo, ma i computer potranno prevedere l'invio di un messaggio in modo casuale, quindi un nodo potrebbe rilasciare il token senza spedire un messaggio.

Per simulare il funzionamento della rete Token ring lo sprite utilizzato per rappresentare le tre tipologie di messaggi dovrà cambiare costume in base alla fase in cui si trova la spedizione (invio del token, del messaggio o della conferma), mentre i quattro computer simuleranno la spedizione o ricezione di uno dei tre tipi di messaggi, o la decisione di non inviare un messaggio, utilizzando dei fumetti.

Anche in questo caso si potranno prevedere due fasi logicamente distinte, come nella prima simulazione, una di trasmissione e una di ricezione, nelle quali sono previste le medesime possibilità già viste in precedenza.

La differenza sostanziale in questa simulazione è la schematizzazione della trasmissione di un messaggio che in questo caso deve prevedere la possibilità che il nodo non abbia dati da trasmettere.



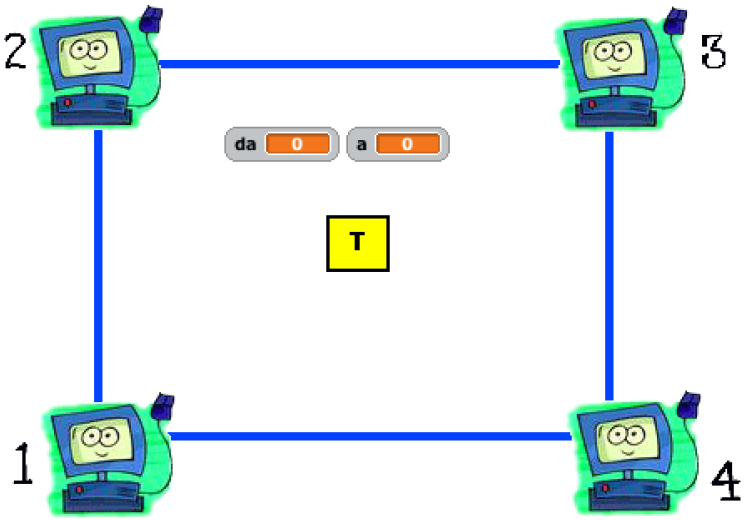
Implementazione della seconda simulazione

Viene di seguito fornita una possibile implementazione di questa seconda versione.

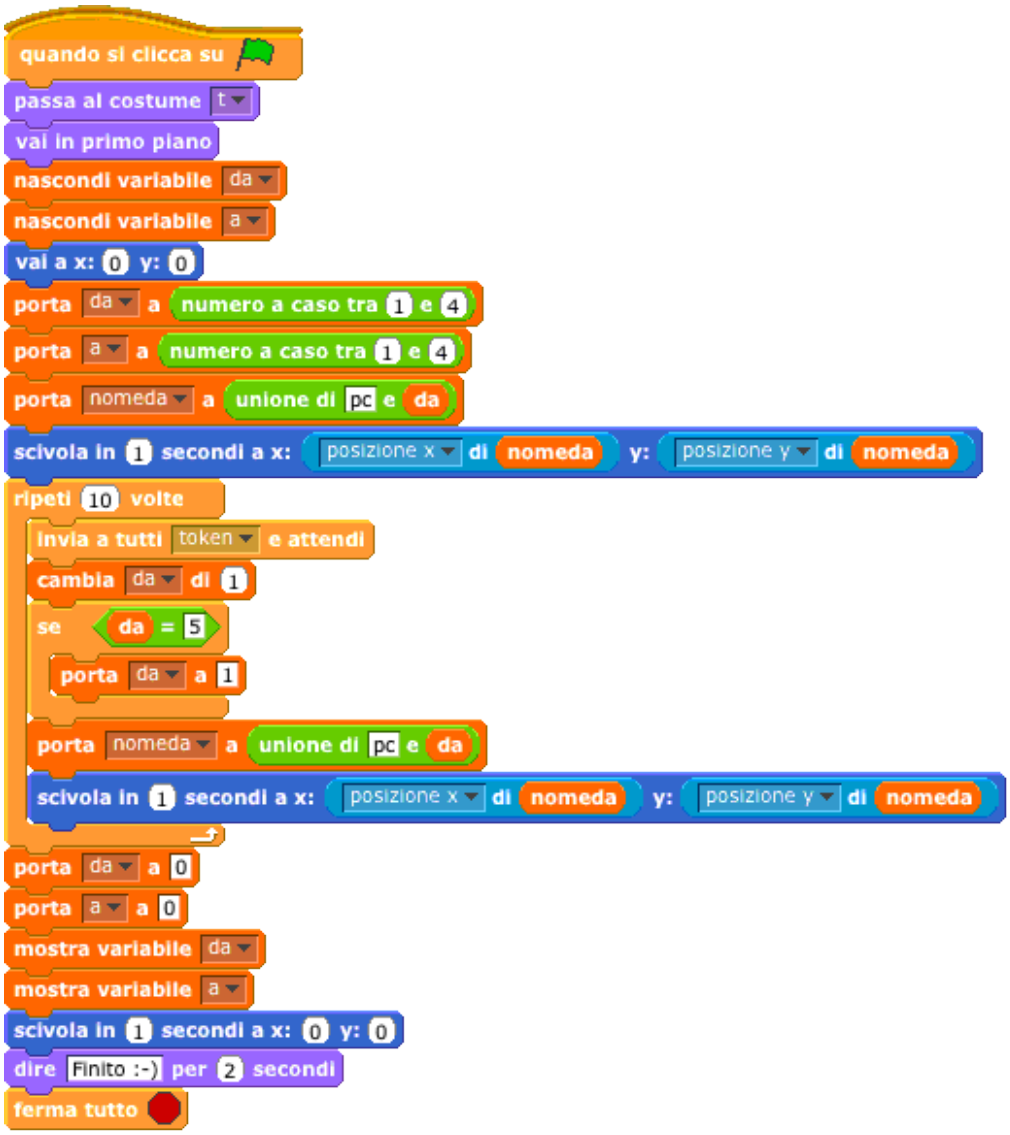
Sono stati previsti:

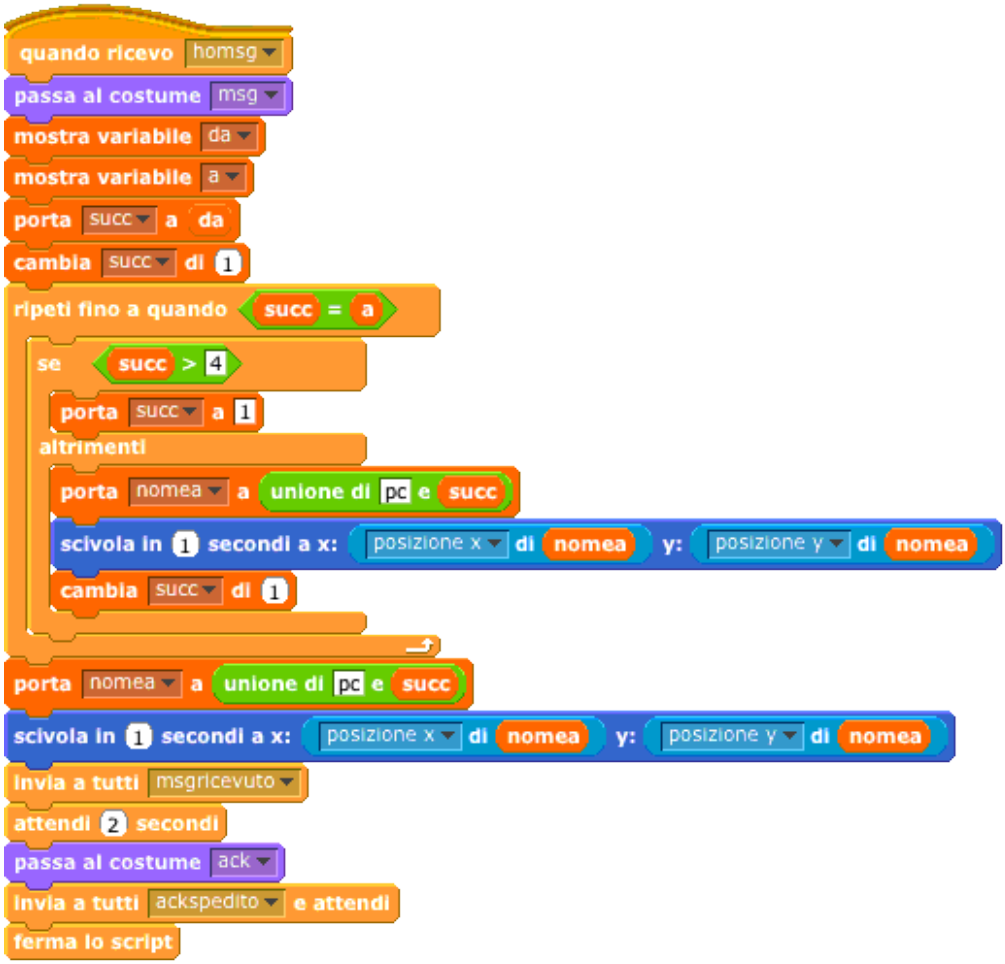
- uno sprite *token* che assumerà i tre costumi dei diversi messaggi;
- uno sprite per ogni computer (*pc1*, *pc2*, *pc3*, *pc4*);
- uno sprite per i singoli numeri associati ai computer;
- la variabile *da* che conterrà il numero del computer mittente;
- la variabile *a* che conterrà il numero del computer destinatario, selezionato in modo casuale;
- la variabile *succ* che conterrà il numero del computer logicamente successivo durante la fase di movimento dello sprite *token* fra i vari computer;
- la variabile *nomeda* che conterrà il nome del computer mittente;
- la variabile *nomea* che conterrà il nome del computer destinatario;
- la variabile *scegli* che permetterà ad ogni singolo sprite che identifica un computer di scegliere casualmente se spedire o meno un messaggio al ricevimento del token;
- il messaggio *token* inviato dallo sprite *token* a tutti gli sprite che identificano un computer per segnalare che uno di loro è in possesso del token; questo messaggio attiverà in ogni sprite computer la scelta casuale di spedire o meno un messaggio, e in caso positivo la selezione casuale di un computer destinatario fra i tre rimanenti con la relativa segnalazione dello stato tramite un fumetto. Nel caso in cui il computer scegliesse di non inviare un messaggio verrebbe esplicitato, anche in questo caso tramite un fumetto;
- il messaggio *homsg* inviato dallo sprite computer, che è entrato in possesso del token, allo sprite *token*, per segnalargli l'intenzione di spedire un messaggio; lo sprite *token* avvierà di conseguenza la gestione del suo scorrimento fra i vari computer per giungere a destinazione;
- il messaggio *msgricevuto* inviato dallo sprite *token* a tutti gli sprite che identificano un computer per segnalare che uno di loro ha ricevuto un messaggio; questo messaggio attiverà in ogni sprite computer una verifica per determinare se si è i destinatari del messaggio e la relativa visualizzazione di un fumetto che esplicita la situazione;
- il messaggio *ackspedito* inviato dallo sprite *token* a se stesso per gestire gli spostamenti dello sprite associati alla fase di spedizione di una conferma;
- il messaggio *ackricevuto* inviato dallo sprite *token* a tutti gli sprite che identificano un computer per segnalare che uno di loro ha ricevuto una conferma; questo messaggio attiverà in ogni sprite computer una verifica per determinare se si è i destinatari della conferma e la relativa visualizzazione di un fumetto che esplicita la situazione.

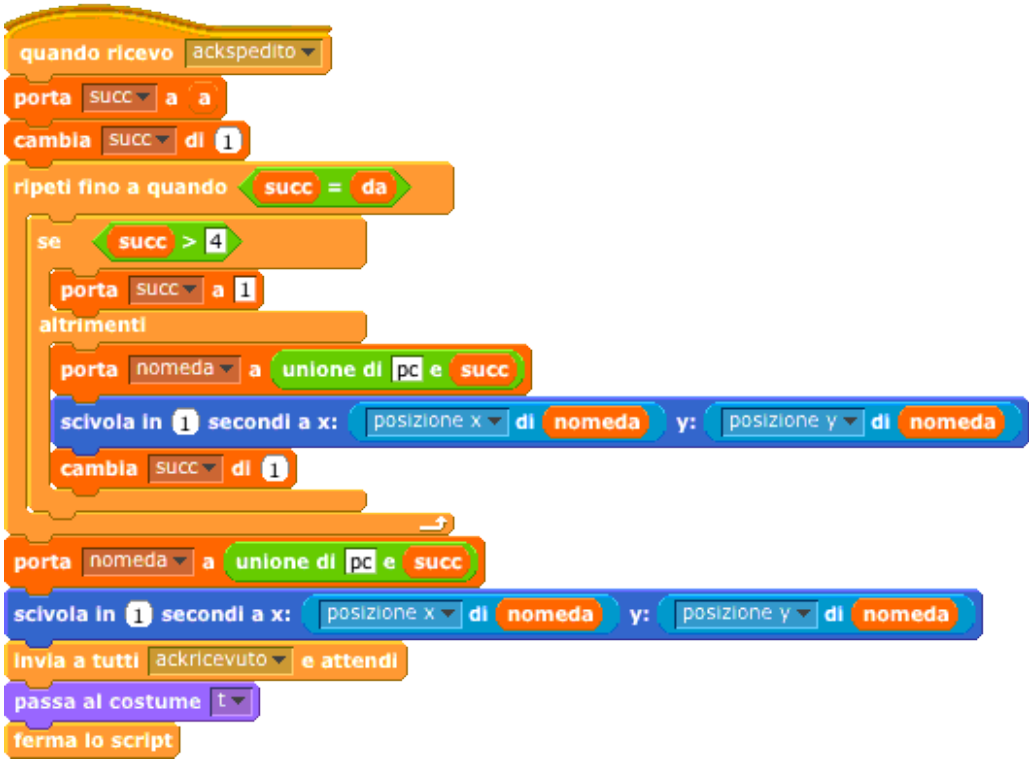
L'ambiente finale sarà il seguente, considerando che le due variabili *da* e *a* verranno visualizzate solo se è effettivamente previsto l'invio di un messaggio:



Script dello sprite *token*:




quando ricevo homsg
passa al costume msg
mostra variabile da
mostra variabile a
porta succ a da
cambia succ di 1
ripeti fino a quando succ = a
se succ > 4
porta succ a 1
altrimenti
porta nomea a unione di pc e succ
scivola in 1 secondi a x: posizione x di nomea y: posizione y di nomea
cambia succ di 1
porta nomea a unione di pc e succ
scivola in 1 secondi a x: posizione x di nomea y: posizione y di nomea
invia a tutti msgricevuto
attendi 2 secondi
passa al costume ack
invia a tutti ackspedito e attendi
ferma lo script


quando ricevo ackspedito
porta succ a a
cambia succ di 1
ripeti fino a quando succ = da
se succ > 4
porta succ a 1
altrimenti
porta nomeda a unione di pc e succ
scivola in 1 secondi a x: posizione x di nomeda y: posizione y di nomeda
cambia succ di 1
porta nomeda a unione di pc e succ
scivola in 1 secondi a x: posizione x di nomeda y: posizione y di nomeda
invia a tutti ackricevuto e attendi
passa al costume t
ferma lo script

Script dello sprite *pc1* (gli script degli sprite *pc2*, *pc3*, *pc4* sono analoghi):