Scratch
Sincronizzazione dei processi

Relazione finale di

Didattica e laboratorio di Calcolatori

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

Scratch
Sincronizzazione di processi

- Scratch e la simulazione di una realtà complessa
- Caso di studio: Token ring
- Implementazione tramite Scratch:
  - Preparazione dell'ambiente
  - Prima simulazione → ogni nodo spedisce un msg
  - Seconda simulazione → il nodo può casualmente decidere di non spedire il msg

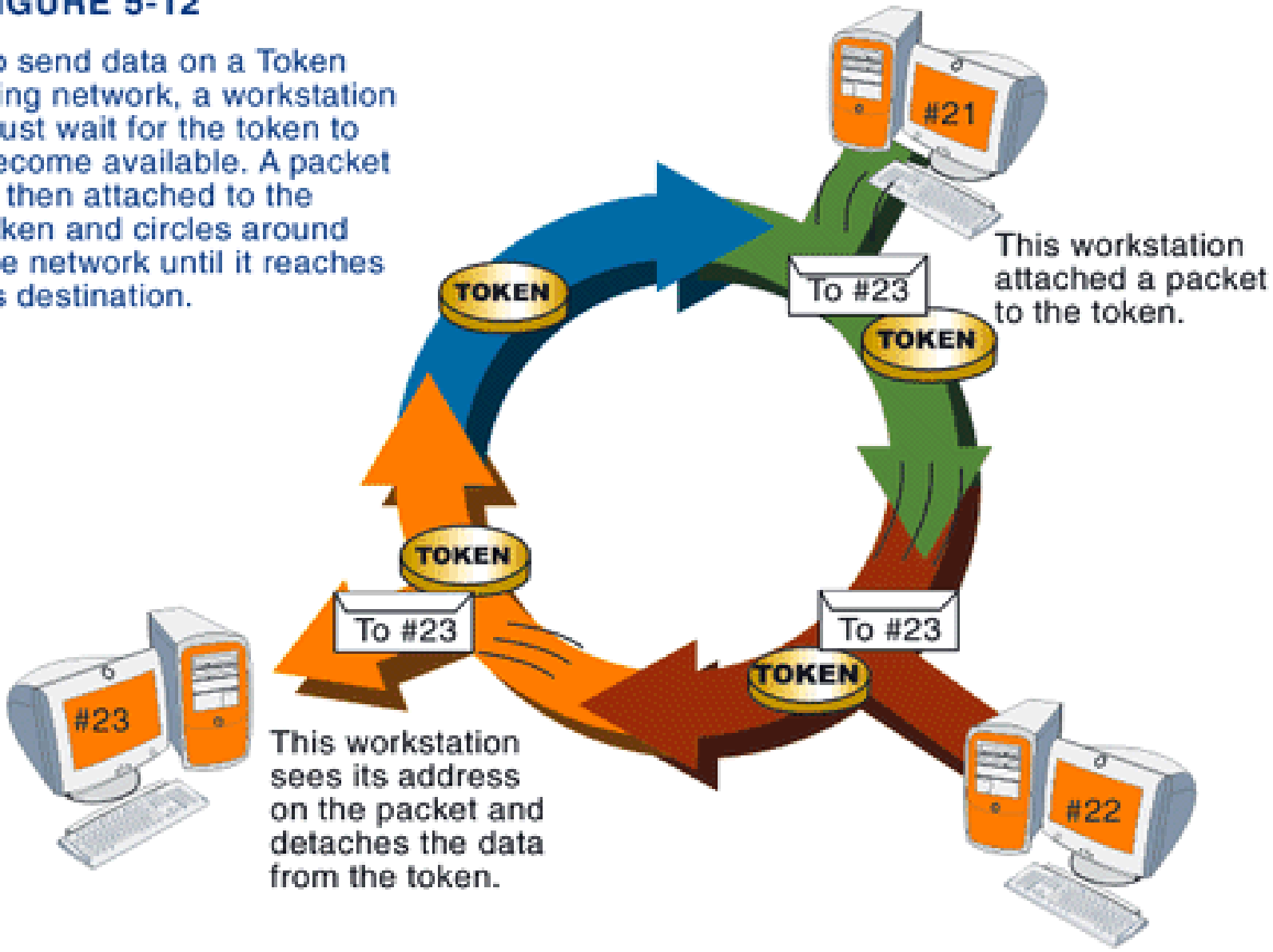
Sratch e la simulazione di una realtà complessa

- Permette di soffermarsi sui concetti logici principali in quanto privo di errori sintattici
- Simulare una realtà obbliga alla sua comprensione
- Destinatari del progetto:
  - Classi seconde → Tecnologie Informatiche
  - Classi del triennio → Informatica

Token ring

FIGURE 5-12

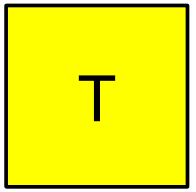
To send data on a Token Ring network, a workstation must wait for the token to become available. A packet is then attached to the token and circles around the network until it reaches its destination.



Scratch: preparazione dell'ambiente

Tre tipologie di messaggi → tre costumi diversi per lo sprite *token*:

- token



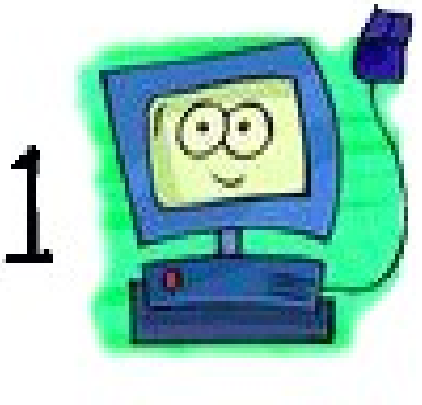
- messaggio

- conferma

Scratch: preparazione dell'ambiente

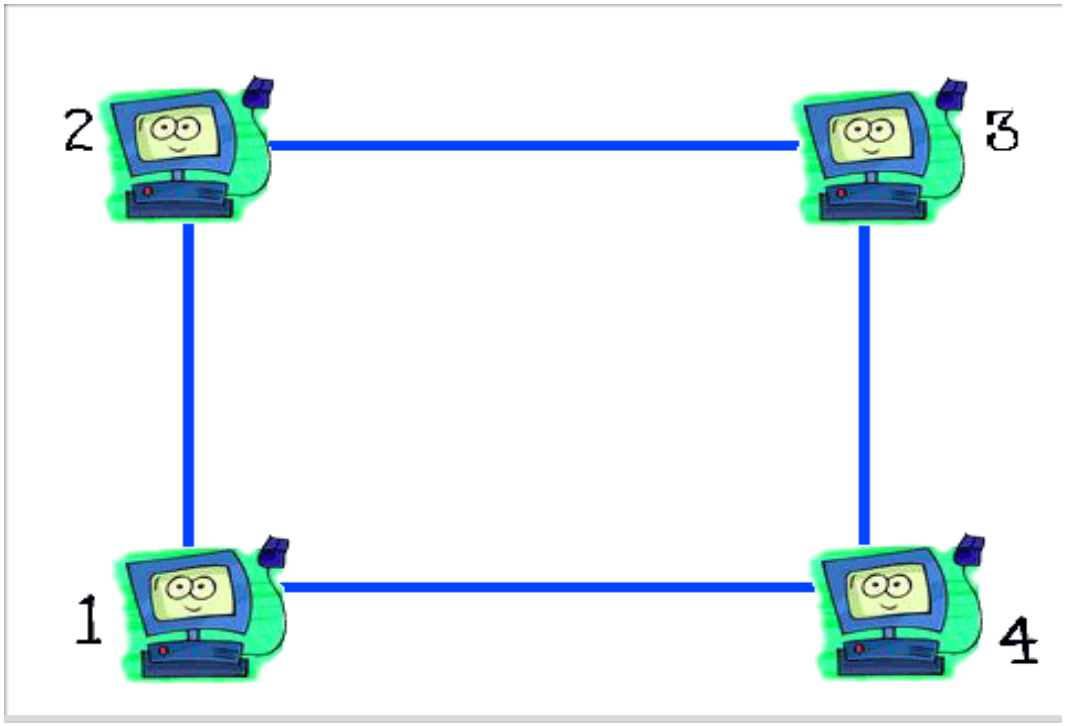
Computer collegati per simulare l'anello:

- quattro sprite *pc*`n` distinti, uno per computer
- quattro sprite con un numero ognuno associato ad un computer



Scratch: preparazione dell'ambiente

Preparare lo stage simulando un anello di computer



Prima simulazione

Ogni nodo spedisce un messaggio

Variabili:

- *da*: numero computer mittente
- *a*: numero computer destinatario
- *succ*: numero computer successivo lungo il cammino
- *nomeda*: nome sprite del computer mittente
- *nomea*: nome sprite del computer destinatario

Prima simulazione

Ogni nodo spedisce un messaggio

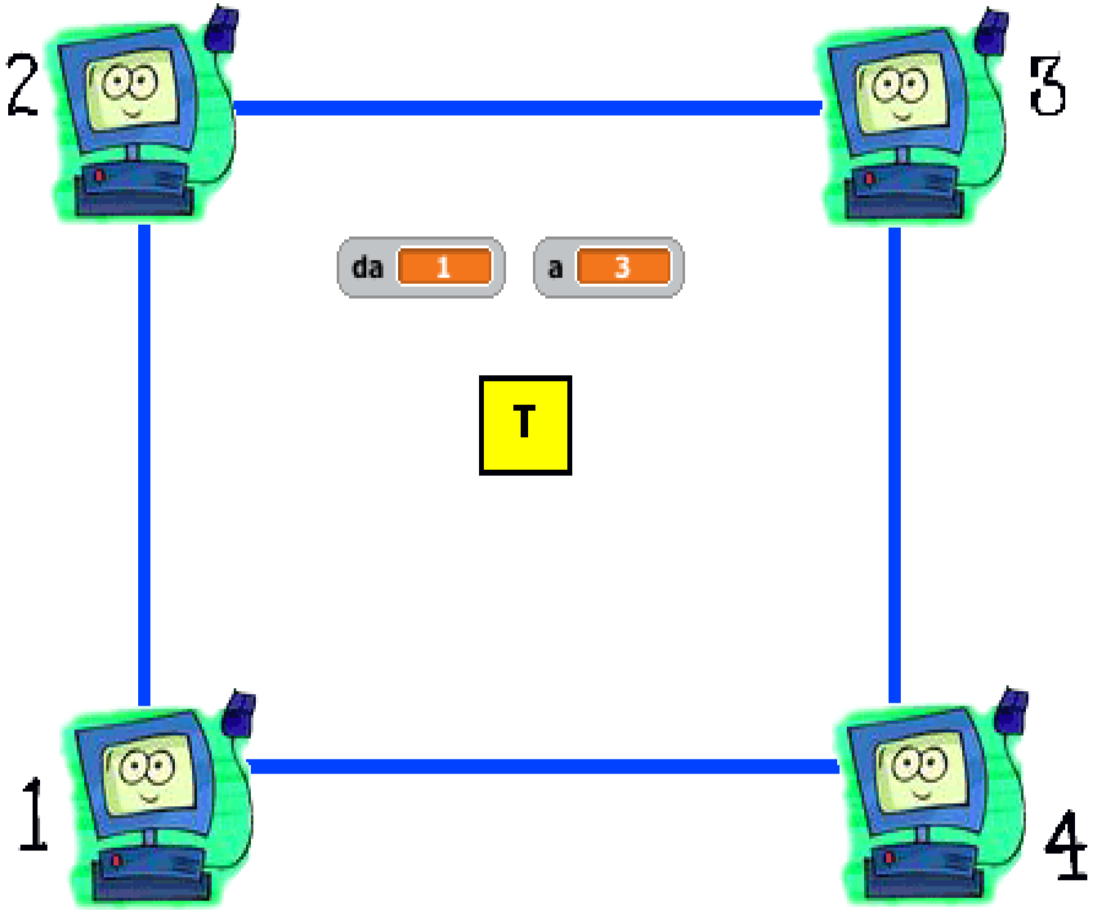
Messaggi scambiati fra gli sprite:

- *token*: *token* → *pc*$_n$, il *pc*$_n$ ha il token, spedisce un msg, segnala con fumetto
- *msgricevuto*: *token* → *pc*$_n$, il *pc*$_n$ ha ricevuto un msg, segnala con fumetto
- *ackspedito*: *token* → *token*, gestione scorrimento fra i *pc*$_n$ per raggiungere la destinazione
- *ackricevuto*: *token* → *pc*$_n$, il *pc*$_n$ ha ricevuto un ack, segnala con fumetto

Prima simulazione

Ogni nodo spedisce un messaggio

Ambiente finale



Prima simulazione

Ogni nodo spedisce un messaggio

Andiamo a vedere gli script:

- sprite *token*
- sprite *pc*`n`

Seconda simulazione

Il nodo può decidere di non spedire il msg

Variabili:

- *da*: numero computer mittente
- *a*: numero computer destinatario
- *succ*: numero computer successivo lungo il cammino
- *nomeda*: nome sprite del computer mittente
- *nomea*: nome sprite del computer destinatario
- ***scegli***: usata dagli sprite *pc*$_n$ per decidere casualmente se spedire o meno un messaggio

Seconda simulazione

Il nodo può decidere di non spedire il msg

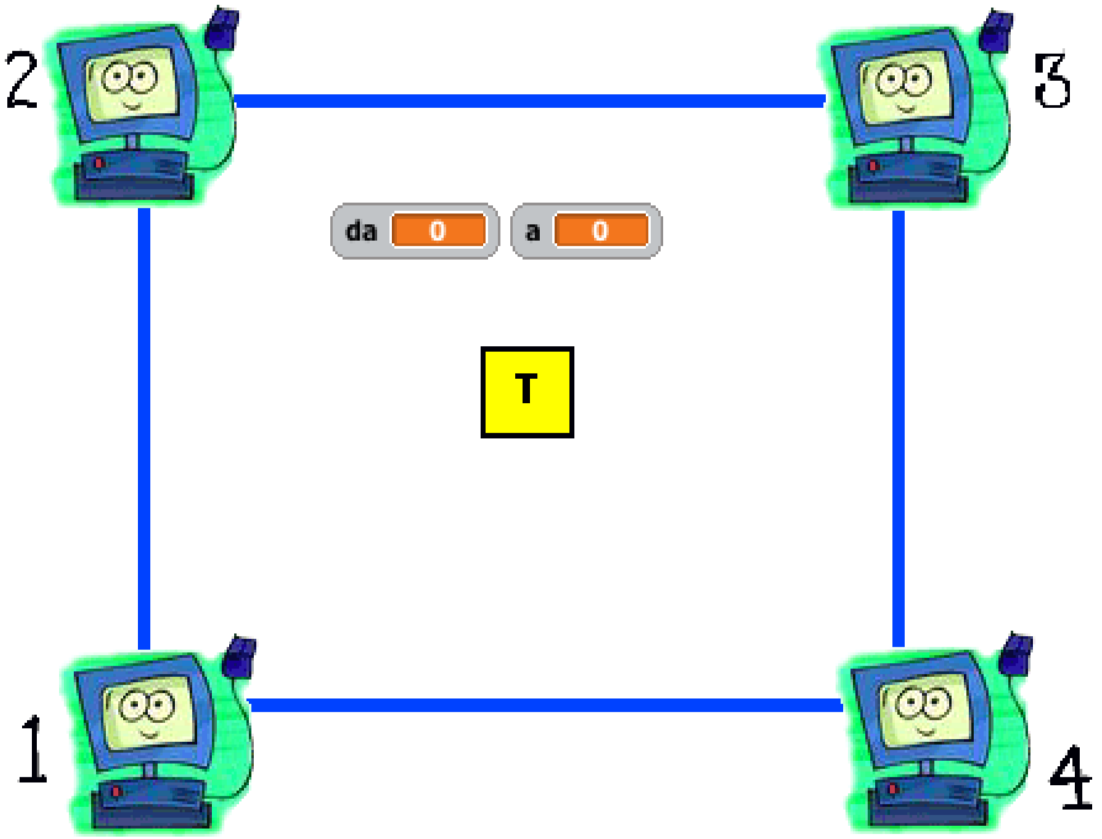
Messaggi scambiati fra gli sprite:

- *token*: *token* → *pc*$_n$, il *pc*$_n$ ha il token **può eventualmente spedire un msg**
- ***homsg***: *pc*$_n$ → *token*, il *pc*$_n$ vuole spedire un msg
- *msgricevuto*: *token* → *pc*$_n$, il *pc*$_n$ ha ricevuto un msg, segnala con fumetto
- *ackspedito*: *token* → *token*, gestione scorrimento fra i *pc*$_n$ per raggiungere la destinazione
- *ackricevuto*: *token* → *pc*$_n$, il *pc*$_n$ ha ricevuto un ack, segnala con fumetto

Seconda simulazione

Il nodo può decidere di non spedire il msg

Ambiente finale



Seconda simulazione

Il nodo può decidere di non spedire il msg

Andiamo a vedere gli script:

- sprite *token*
- sprite *pc*`n`

Finito