Concurrent Maintenance of Rings

Xiaozhou Li Jayadev Misra C. Greg Plaxton

March 6, 2004

Structured Peer-to-Peer Networks

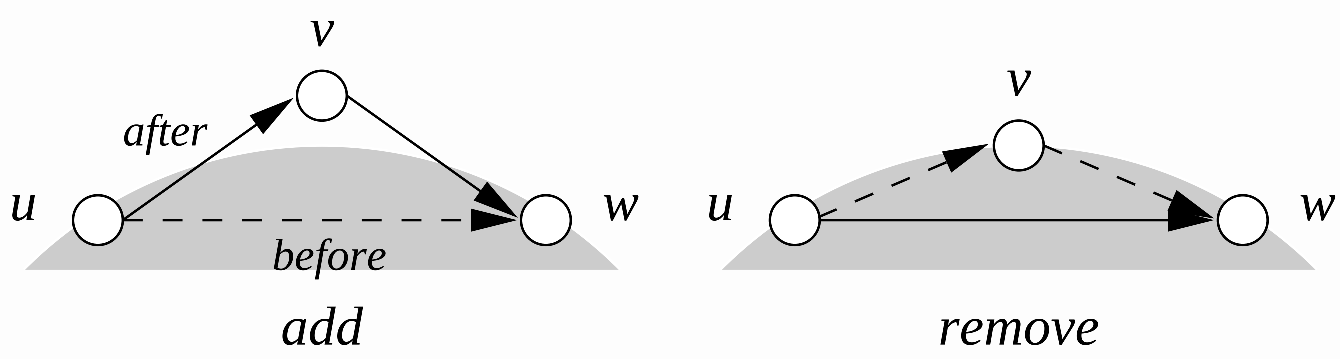
- Nodes have neighbor variables
- The neighbor variables collectively form a certain topology: ring, hypercube, etc
- Over time, nodes may join or leave, possibly concurrently
- The neighbor variables should be properly updated to maintain the topology
- Problem: Design, and prove the correctness of, protocols that maintain the topology
- Focus of this paper: Ring topology

Maintenance of Rings

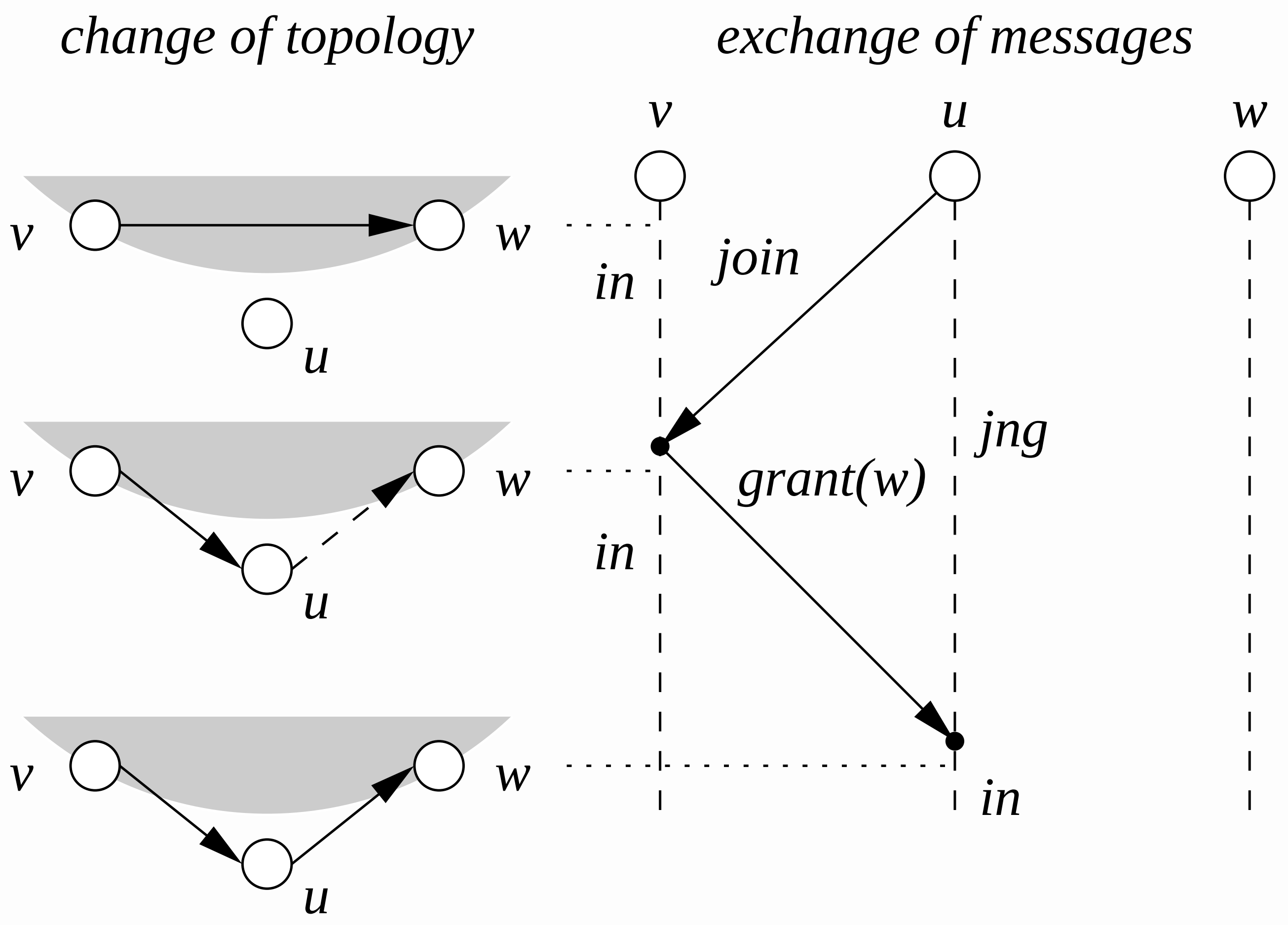
- Joins for a unidirectional ring
- Joins for a bidirectional ring
- Leaves for a bidirectional ring
- Joins and leaves for a bidirectional ring
- Joins and leaves for multiple rings

Preliminaries

- $ring(x) = \langle \forall u, v : u.x \neq \textbf{nil} \wedge v.x \neq \textbf{nil} : u \overset{x}{\hookrightarrow} v \rangle$, where $u \overset{x}{\hookrightarrow} v = \langle \exists i : i > 0 : u.x^i = v \rangle$
- **Lemma** In a ring, distinct processes have distinct neighbors.
- **Lemma**



Joins for a Unidirectional Ring



The Protocol

process p
var $s : \{in, out, jng\};$ {state}
$r : V';$ {right neighbor}
$a : V'$ {auxiliary variable}
init $s = out \wedge r = \textbf{nil}$
begin
$\square\ s = out \rightarrow \{T_1\}$
$a := contact();$
if $a = p \rightarrow r, s := p, in$
$\square\ a \neq p \rightarrow s := jng;$ **send** $join()$ **to** a **fi**
$\square$ **rcv** $join()$ **from** $q \rightarrow \{T_2\}$
if $s = in \rightarrow$ **send** $grant(r)$ **to** $q;\ r := q$
$\square\ s \neq in \rightarrow$ **send** $retry()$ **to** q **fi**
$\square$ **rcv** $grant(a)$ **from** $q \rightarrow \{T_3\}$
$r, s := a, in$
$\square$ **rcv** $retry()$ **from** $q \rightarrow \{T_4\}$
$s := out$
end

An Invariant

- $ring(r)$?
- Define $u.r'$ as:

$$u.r' = \begin{cases} x & \text{if } m^{-}(grant, u) = 1 \wedge \\ & \quad m^{-}(grant(x), u) = 1 \\ u.r & \text{otherwise.} \end{cases}$$

- $m^{-}(msg, u)$: number of incoming messages of type msg of u
- $ring(r')$?

The Invariant

- $I = A \wedge B \wedge C \wedge ring(r')$
- $A = \langle \forall u :: (u.s = jng \equiv f(u) = 1) \wedge f(u) \leq 1 \rangle$
- $B = \langle \forall u :: u.s = in \equiv u.r \neq \mathbf{nil} \rangle$
- $C = \#grant(\mathbf{nil}) = 0$
- $f(u) = m^{+}(join, u) + m^{-}(grant, u) + m^{-}(retry, u)$
- $\#grant(\mathbf{nil})$: number of *grant* messages with parameter **nil** in all channels

Theorems and Proofs

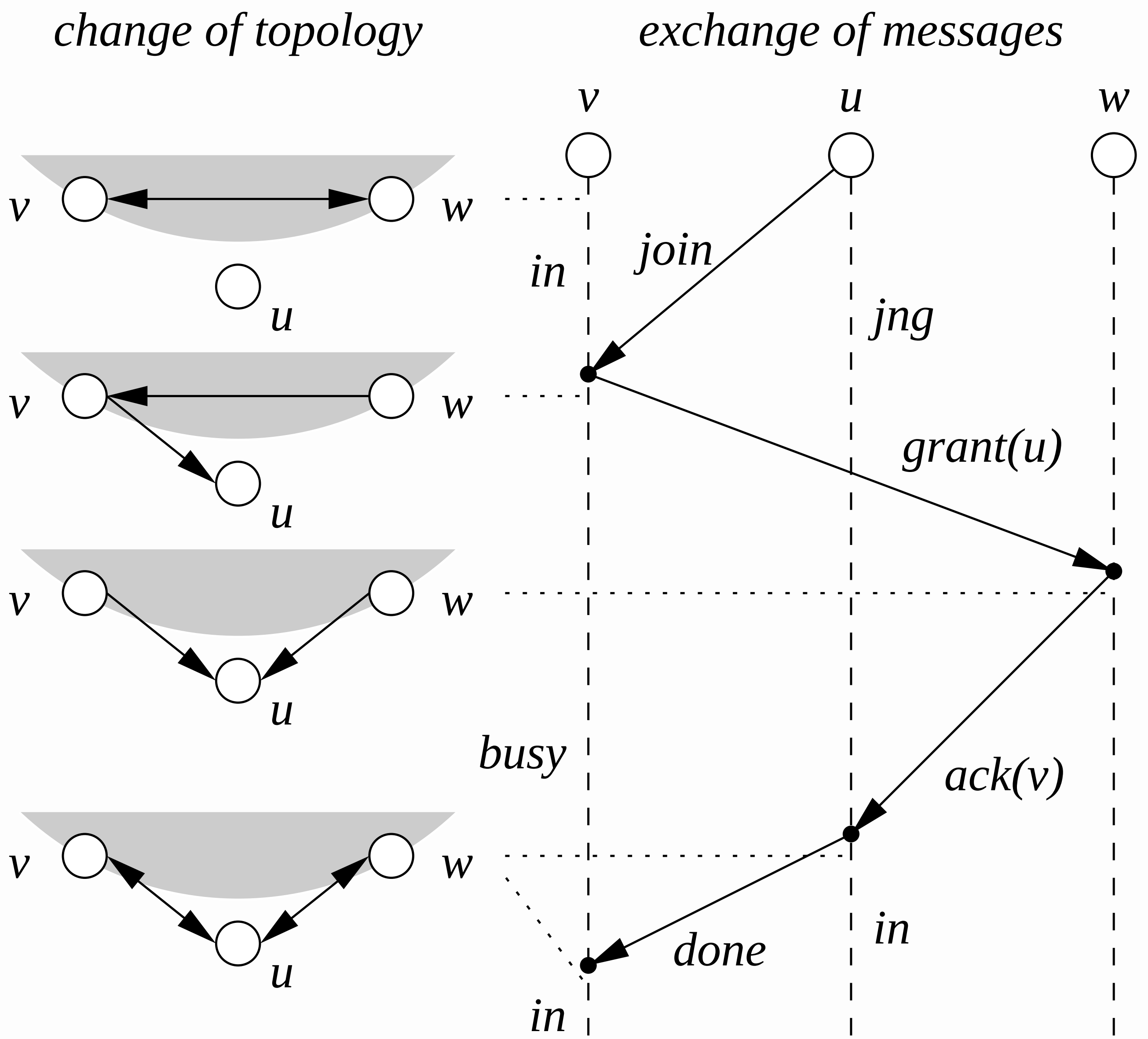
- **Theorem** I is an invariant.
- Proof: Check that every conjunct is preserved by every action
- **Theorem** If joins eventually subside, then $ring(r)$ eventually holds, and once joins subside, $ring(r)$ is stable.

Excerpt of a Proof

$\{ring(r')\}\ T_2\ \{ring(r')\}$: **(**$s = in$**)**

$$
\begin{array}{ll}
 & \uparrow\ p.r = w \wedge p.s = in \wedge m(join, q, p) > 0 \\
\Rightarrow & \quad \{A;\ B;\ \text{def. of } r'\} \\
 & \uparrow\ p.r' = w \wedge m^{-}(grant, p) = 0 \wedge \\
 & \quad\ q.r' = \mathbf{nil} \wedge m^{-}(grant, q) = 0 \\
\Rightarrow & \quad \{\text{action; } p \neq q \text{ because } p.r' \neq q.r' \text{; def. of } r'\} \\
 & \downarrow\ p.r' = q \wedge q.r' = w
\end{array}
$$

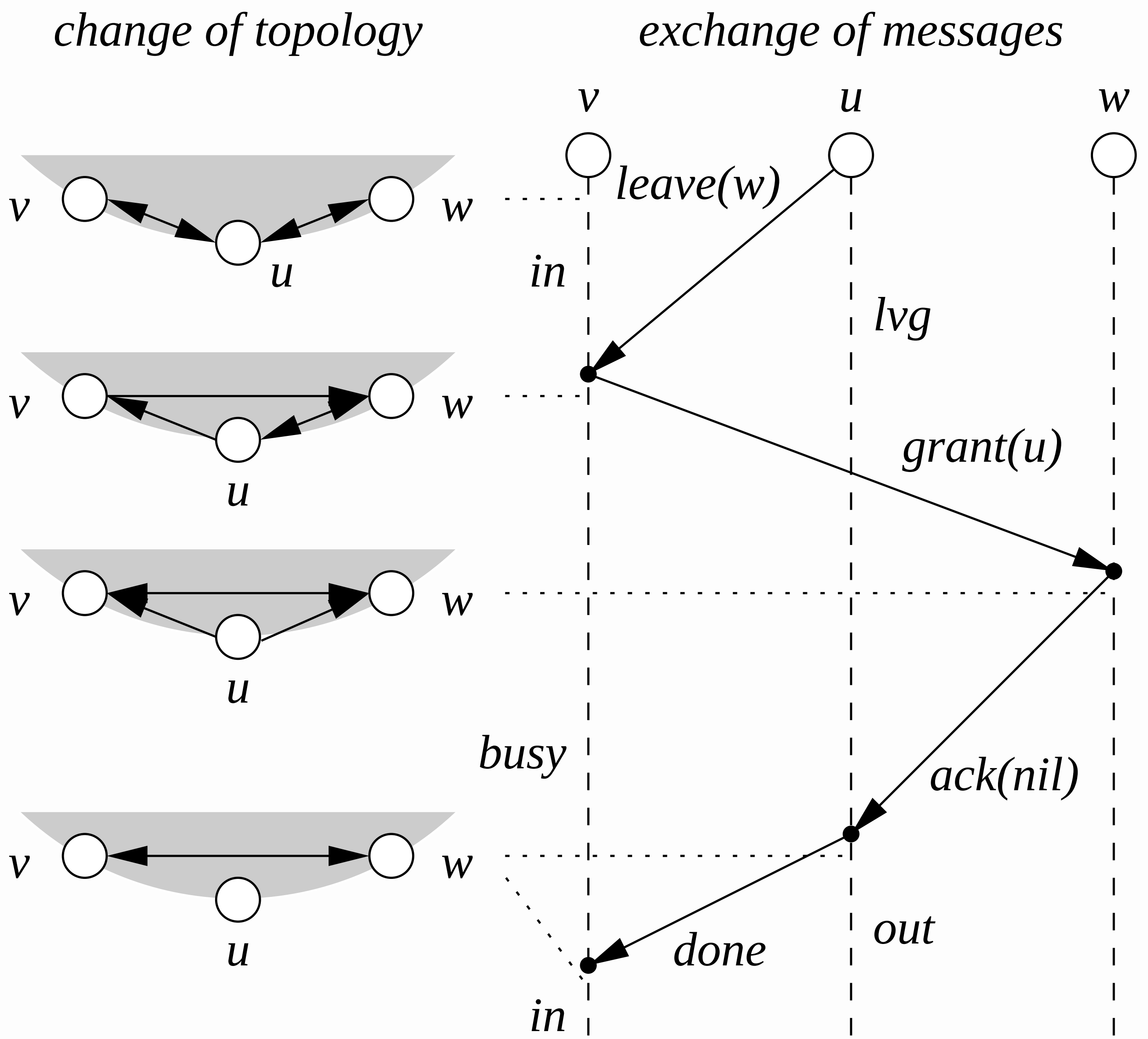
Joins for a Bidirectional Ring



The Join Protocol

process p
var $s : \{in, out, jng, busy\};\ \{\text{state}\}$
$r, l : V';\ \{\text{neighbors}\}$
$t, a : V'\ \{\text{auxiliary variables}\}$
init $s = out \wedge r = \textbf{nil} \wedge l = \textbf{nil} \wedge t = \textbf{nil}$
begin
$\Box\ s = out \rightarrow \{T_1\}$
$a := contact();$
if $a = p \rightarrow r, l, s := p, p, in$
$\Box\ a \neq p \rightarrow s := jng;$ **send** $join()$ **to** a **fi**
$\Box$ **rcv** $join()$ **from** $q \rightarrow \{T_2\}$
if $s = in \rightarrow$ **send** $grant(q)$ **to** $r;\ r, s, t := q, busy, r$
$\Box\ s \neq in \rightarrow$ **send** $retry()$ **to** q **fi**
$\Box$ **rcv** $grant(a)$ **from** $q \rightarrow \{T_3\}$
send $ack(l)$ **to** $a;\ l := a$
$\Box$ **rcv** $ack(a)$ **from** $q \rightarrow \{T_4\}$
$r, l, s := q, a, in;$ **send** $done()$ **to** l
$\Box$ **rcv** $done()$ **from** $q \rightarrow \{T_5\}$
$s, t := in,$ **nil**
$\Box$ **rcv** $retry()$ **from** $q \rightarrow \{T_6\}$
$s := out$
end

Leaves for a Bidirectional Ring



The Leave Protocol

process p

var $s : \{in, out, lvg, busy\};$ {state}

$r, l : V';$ {neighbors}

$t, a : V'$ {auxiliary variables}

init $s = out \wedge r = \mathbf{nil} \wedge l = \mathbf{nil} \wedge t = \mathbf{nil}$

begin

$\Box\ s = in \rightarrow \{T_1\}$

if $l = p \rightarrow r, l, s := \mathbf{nil}, \mathbf{nil}, out$

$\Box\ l \neq p \rightarrow s := lvg;$ **send** $leave(r)$ **to** l **fi**

$\Box$ **rcv** $leave(a)$ **from** $q \rightarrow \{T_2\}$

if $s = in \wedge r = q \rightarrow$ **send** $grant(q)$ **to** $a;\ r, s, t := a, busy, r$

$\Box\ s \neq in \vee r \neq q \rightarrow$ **send** $retry()$ **to** q **fi**

$\Box$ **rcv** $grant(a)$ **from** $q \rightarrow \{T_3\}$

send $ack(\mathbf{nil})$ **to** $a;\ l := q$

$\Box$ **rcv** $ack(a)$ **from** $q \rightarrow \{T_4\}$

send $done()$ **to** $l;\ r, l, s := \mathbf{nil}, \mathbf{nil}, out$

$\Box$ **rcv** $done()$ **from** $q \rightarrow \{T_5\}$

$s, t := in, \mathbf{nil}$

$\Box$ **rcv** $retry()$ **from** $q \rightarrow \{T_6\}$

$s := in$

end

The Combined Protocol

process p
var $s : \{in, out, jng, lvg, busy\};$ {state}
$r, l : V';$ {neighbors}
$t, a : V'$ {auxiliary variables}
init $s = out \wedge r = \textbf{nil} \wedge l = \textbf{nil} \wedge t = \textbf{nil}$
begin
□ $s = out \rightarrow \{T_1^j\}$ $a := contact();$
  if $a = p \rightarrow r, l, s := p, p, in$
  □ $a \neq p \rightarrow s := jng;$ **send** $join()$ **to** a **fi**
□ $s = in \rightarrow \{T_1^l\}$
  if $l = p \rightarrow r, l, s :=$ **nil**, **nil**, out
  □ $l \neq p \rightarrow s := lvg;$ **send** $leave(r)$ **to** l **fi**
□ **rcv** $join()$ **from** $q \rightarrow \{T_2^j\}$
  if $s = in \rightarrow$ **send** $grant(q)$ **to** $r;$ $r, s, t := q, busy, r$
  □ $s \neq in \rightarrow$ **send** $retry()$ **to** q **fi**
□ **rcv** $leave(a)$ **from** $q \rightarrow \{T_2^l\}$
  if $s = in \wedge r = q \rightarrow$ **send** $grant(q)$ **to** $a;$ $r, s, t := a, busy, r$
  □ $s \neq in \vee r \neq q \rightarrow$ **send** $retry()$ **to** q **fi**
□ **rcv** $grant(a)$ **from** $q \rightarrow \{T_3\}$
  if $l = q \rightarrow$ **send** $ack(l)$ **to** $a;$ $l := a$
  □ $l \neq q \rightarrow$ **send** $ack($**nil**$)$ **to** $a;$ $l := q$ **fi**
□ **rcv** $ack(a)$ **from** $q \rightarrow \{T_4\}$
  if $s = jng \rightarrow r, l, s := q, a, in;$ **send** $done()$ **to** l
  □ $s = lvg \rightarrow$ **send** $done()$ **to** $l;$ $r, l, s :=$ **nil**, **nil**, out **fi**
□ **rcv** $done()$ **from** $q \rightarrow \{T_5\}$ $s, t := in,$ **nil**
□ **rcv** $retry()$ **from** $q \rightarrow \{T_6\}$
  if $s = jng \rightarrow s := out$ □ $s = lvg \rightarrow s := in$ **fi**
end

Theorems and Proofs

- Theorems: Similar to those established for the unidirectional join protocol
- Proofs: Define (more involved) r', l', and invariant I and check that every conjunct of I is preserved by every action

Next Step: Machine-Checked Proofs