EXPLOITING LOOPHOLES IN CAP

Michael T. Nygard
Relevance, Inc.

About CAP

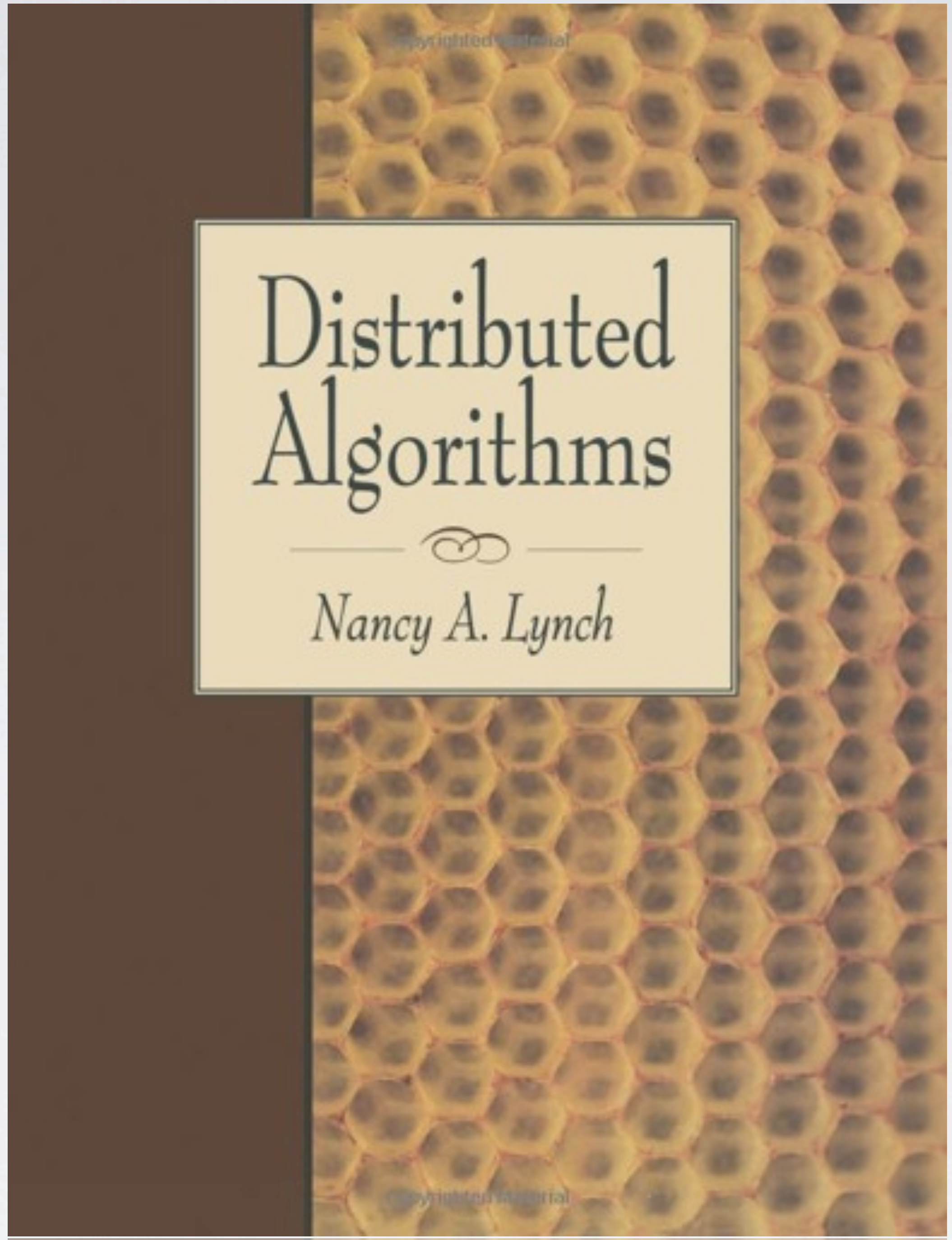
a.k.a. Brewer's Conjecture

a.k.a. Theorem that Shipped 1,000 Launches

"Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services."

Seth Gilbert and Nancy Lynch.

SIGACT News 33, 2 (June 2002), 51-59. DOI=10.1145/564585.564601
http://doi.acm.org/10.1145/564585.564601


Distributed
Algorithms
Nancy A. Lynch

Consistency

Consistency

Availability

Consistency

Availability

Partition-Tolerance

Consistency

Availability

Partition-Tolerance

Choose Two

BEWARE BAD LOGIC

$$C \cap P \rightarrow \neg A$$

BEWARE BAD LOGIC

$$C \cap P \rightarrow \neg A$$

$$A \cap P \rightarrow \neg C$$

BEWARE BAD LOGIC

$$C \cap P \rightarrow \neg A$$

$$A \cap P \rightarrow \neg C$$

$$\nvDash \neg C \rightarrow A$$

CAP

CAP

Gödel's Incompleteness Theorem

CAP

Gödel's Incompleteness Theorem

Heisenberg's Uncertainty Principle

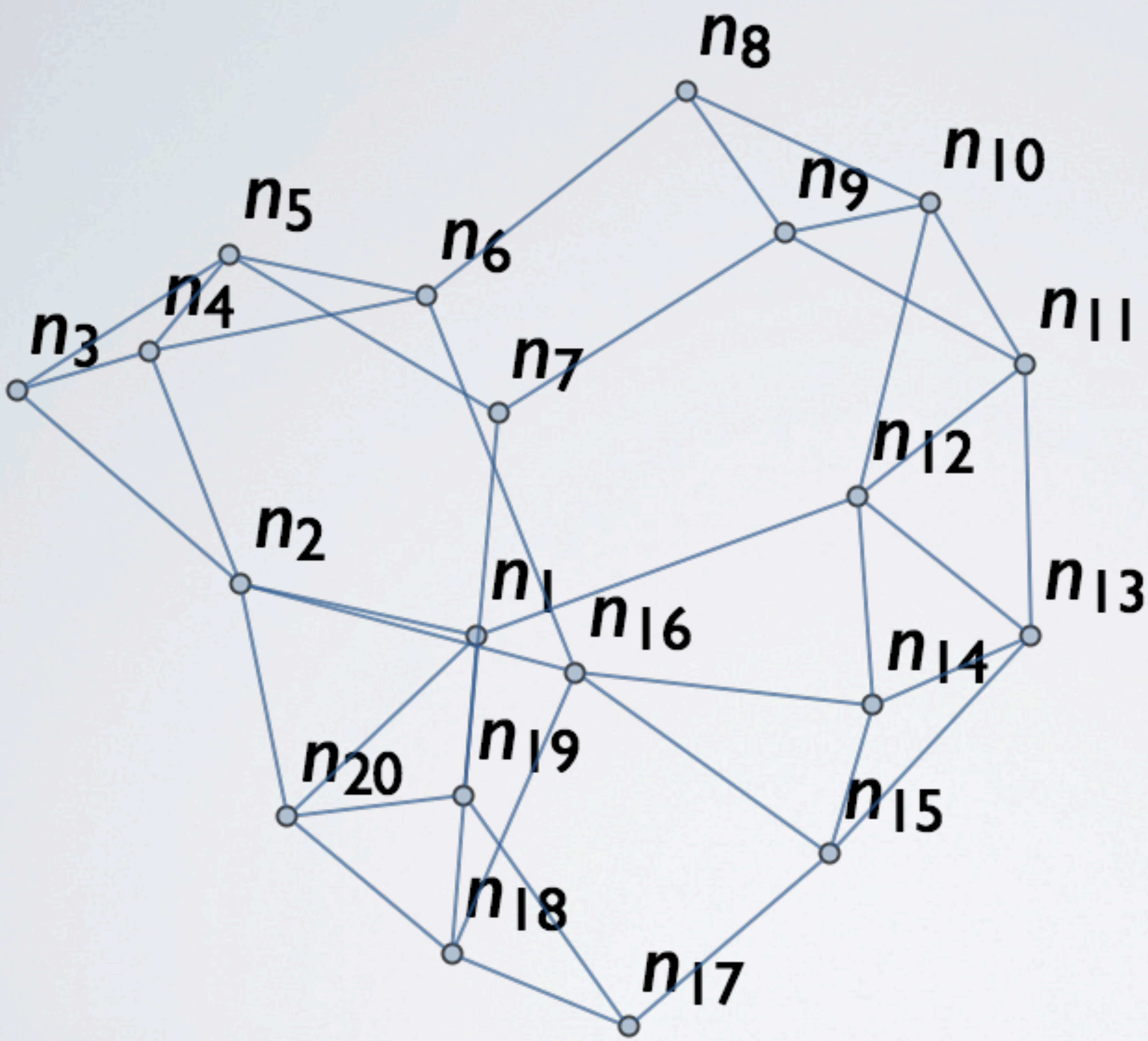
CAP

GIT

HUP

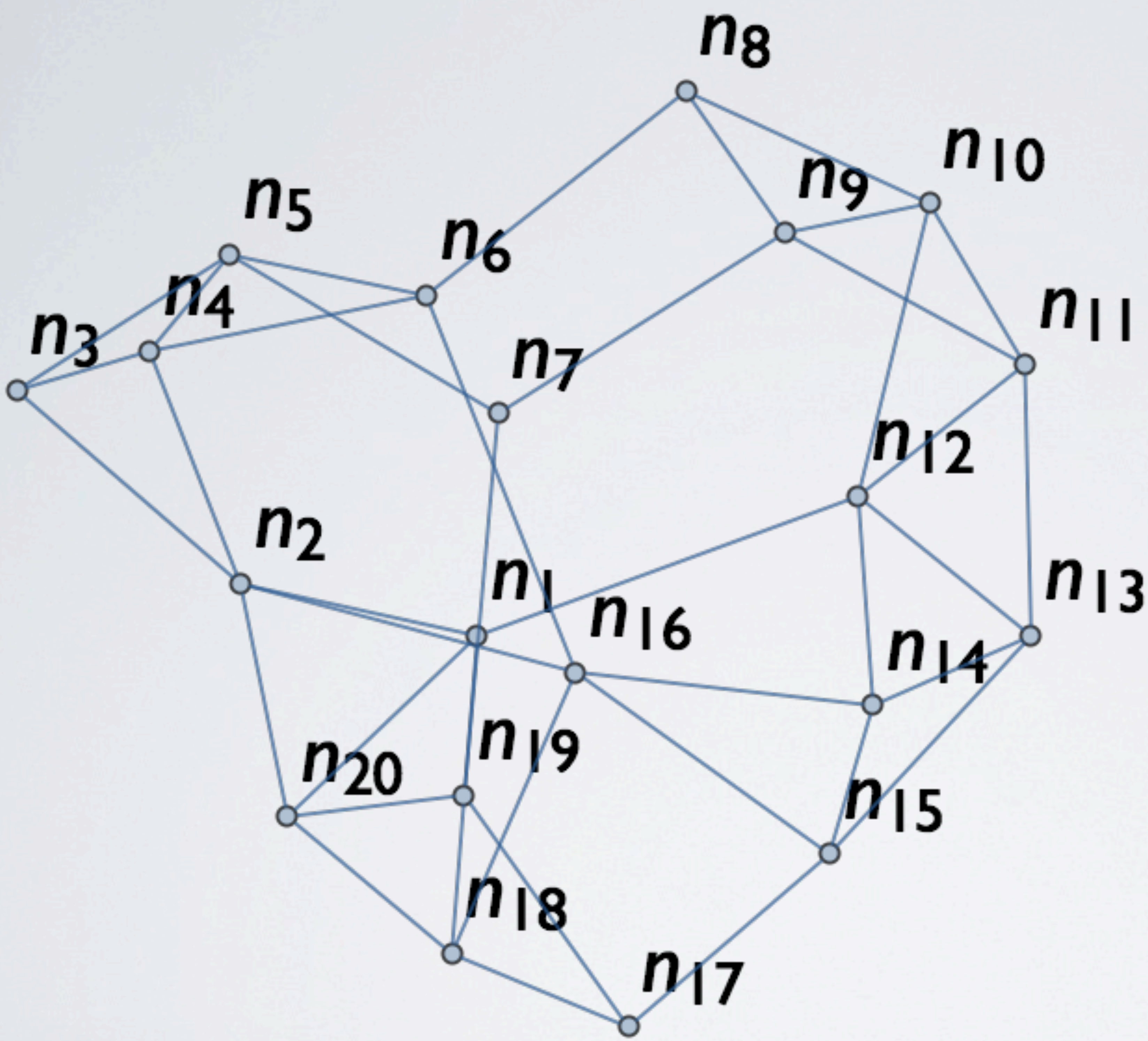
They’re a drag!

The Network



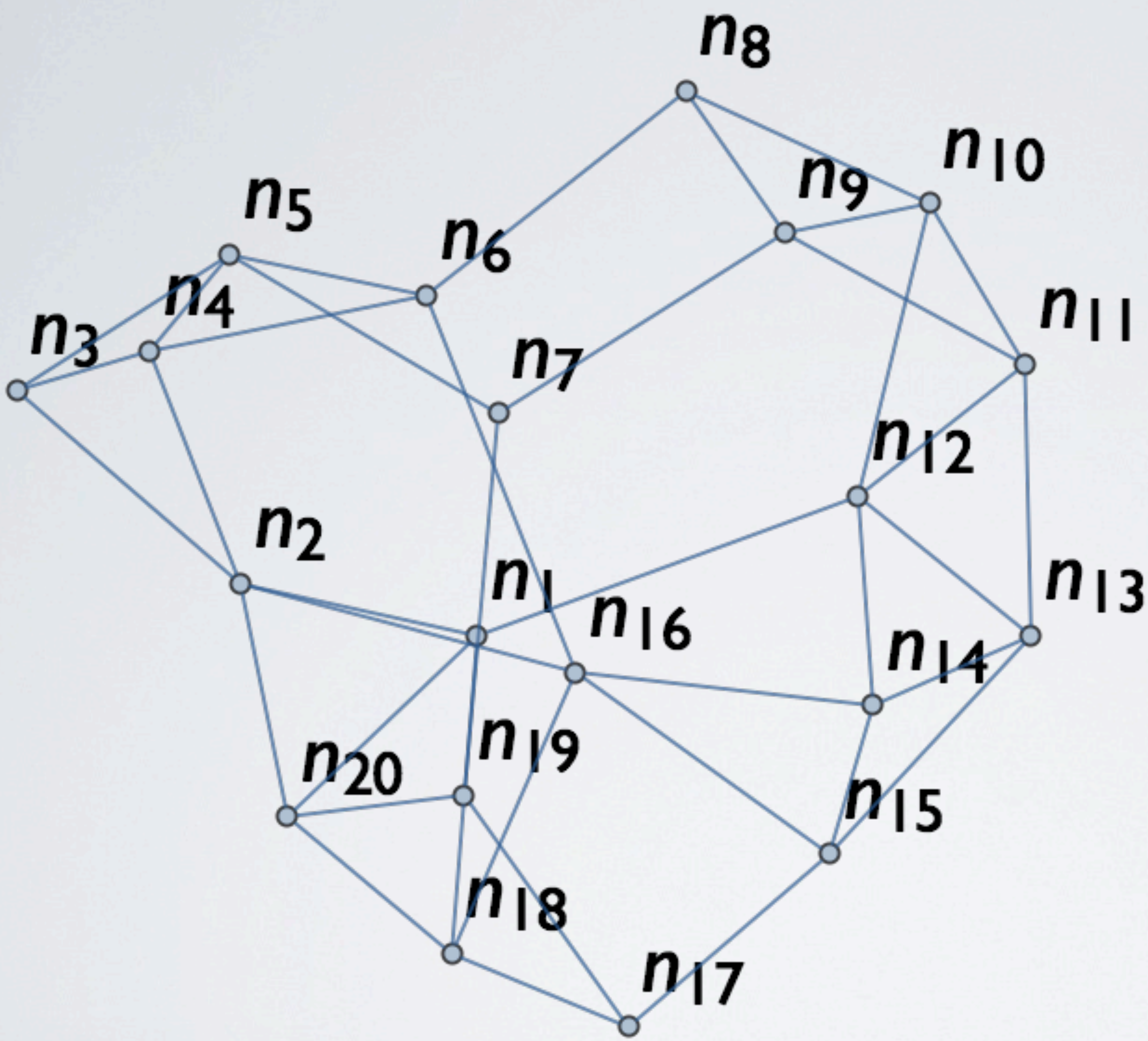
Asynchronous Message-passing Network

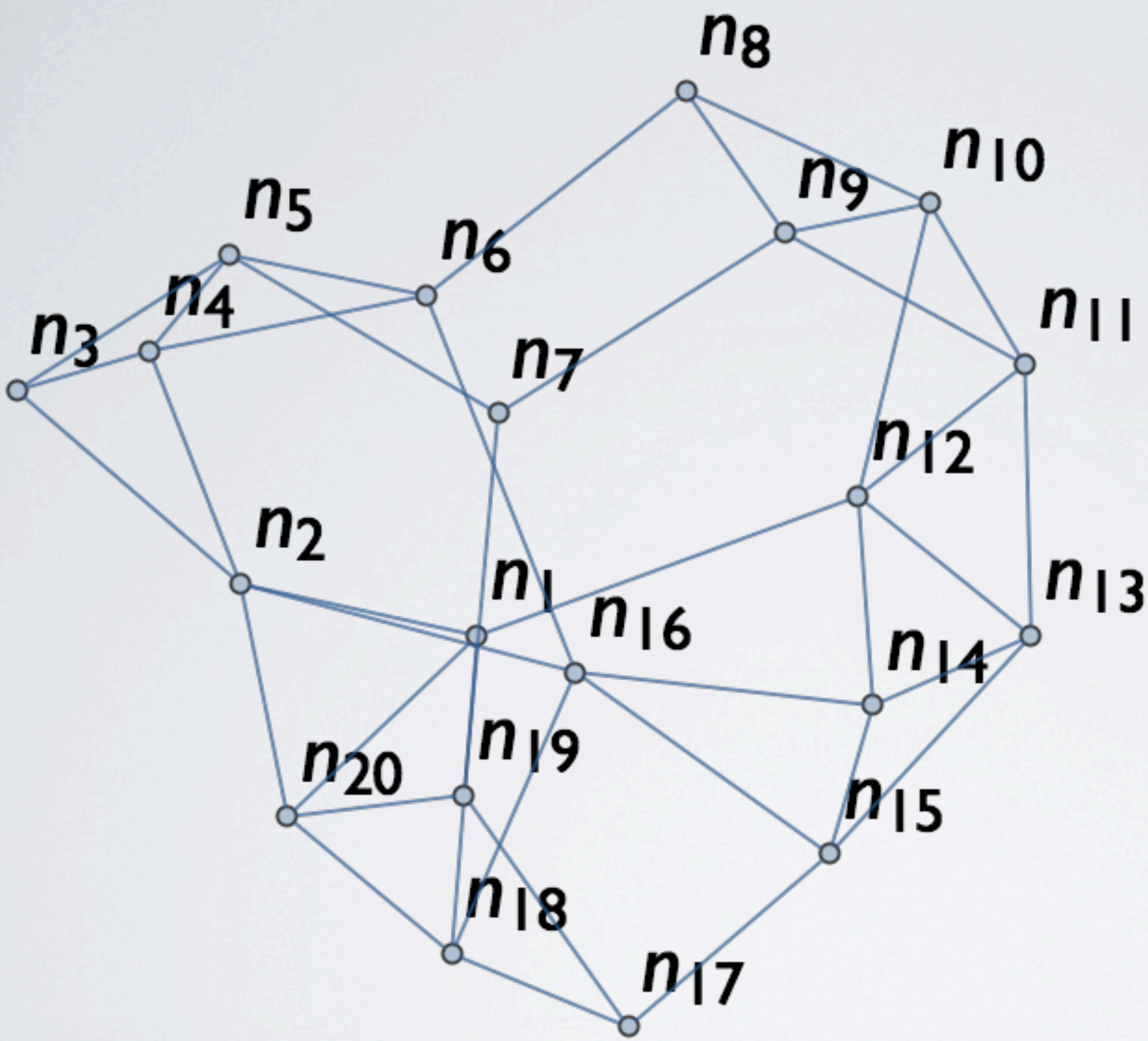
Consistency



n_8
n_{10}
n_9
n_5
n_6
n_4
n_3
n_7
n_{11}
n_{12}
n_2
n_1
n_{16}
n_{13}
n_{14}
n_{20}
n_{19}
n_{15}
n_{18}
n_{17}

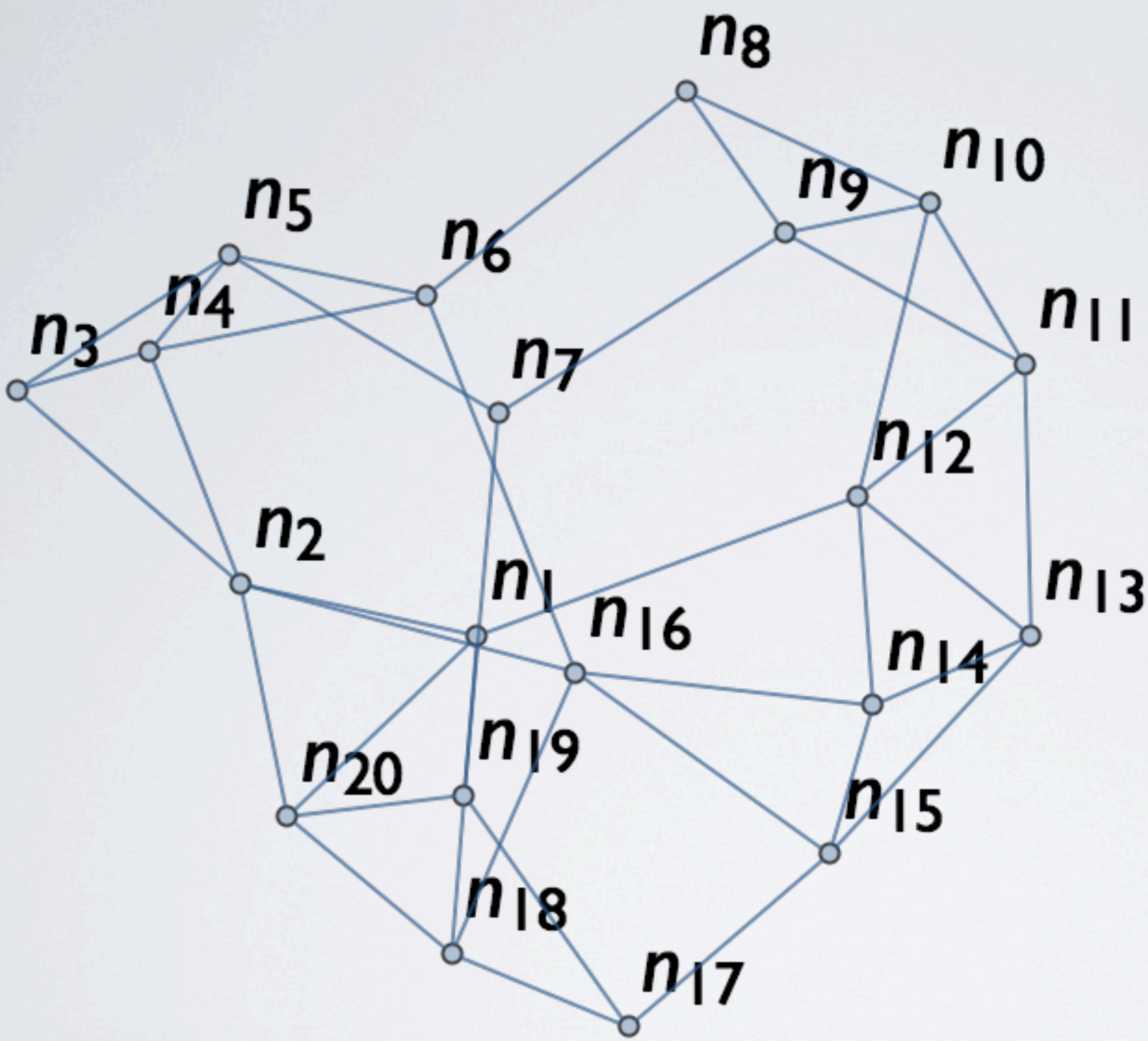
n_2


n_8
n_{10}
n_9
n_5
n_6
n_4
n_3
n_{11}
n_7
n_{12}
n_2
n_{13}
n_1
n_{16}
n_{14}
n_{19}
n_{20}
n_{15}
n_{18}
n_{17}


n_8
n_9
n_{10}
n_5
n_6
n_4
n_3
n_7
n_{11}
n_{12}
n_2
n_1
n_{16}
n_{13}
n_{14}
n_{20}
n_{19}
n_{15}
n_{18}
n_{17}

n_2

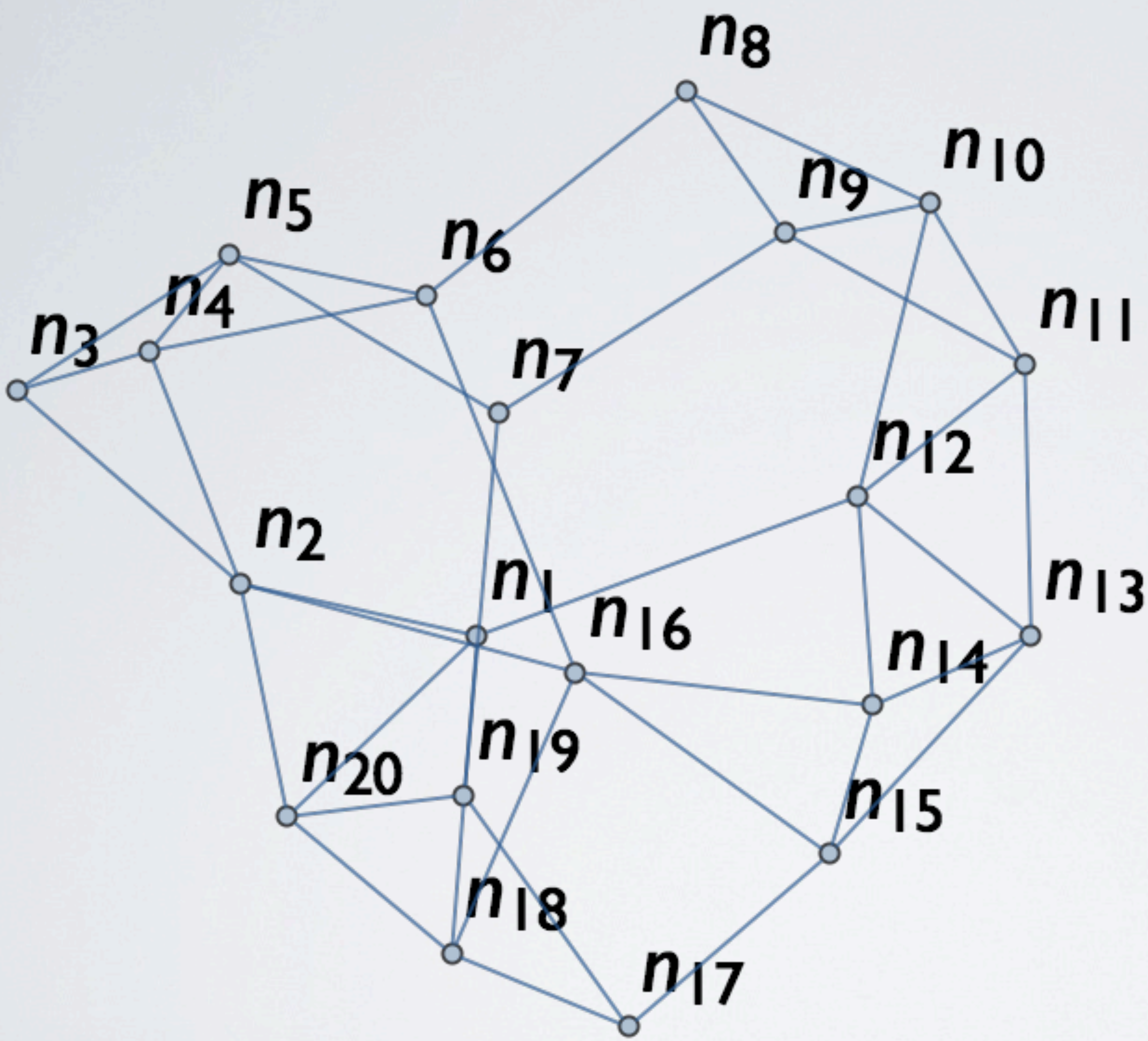
→ write 12



$\boldsymbol{n_2}$

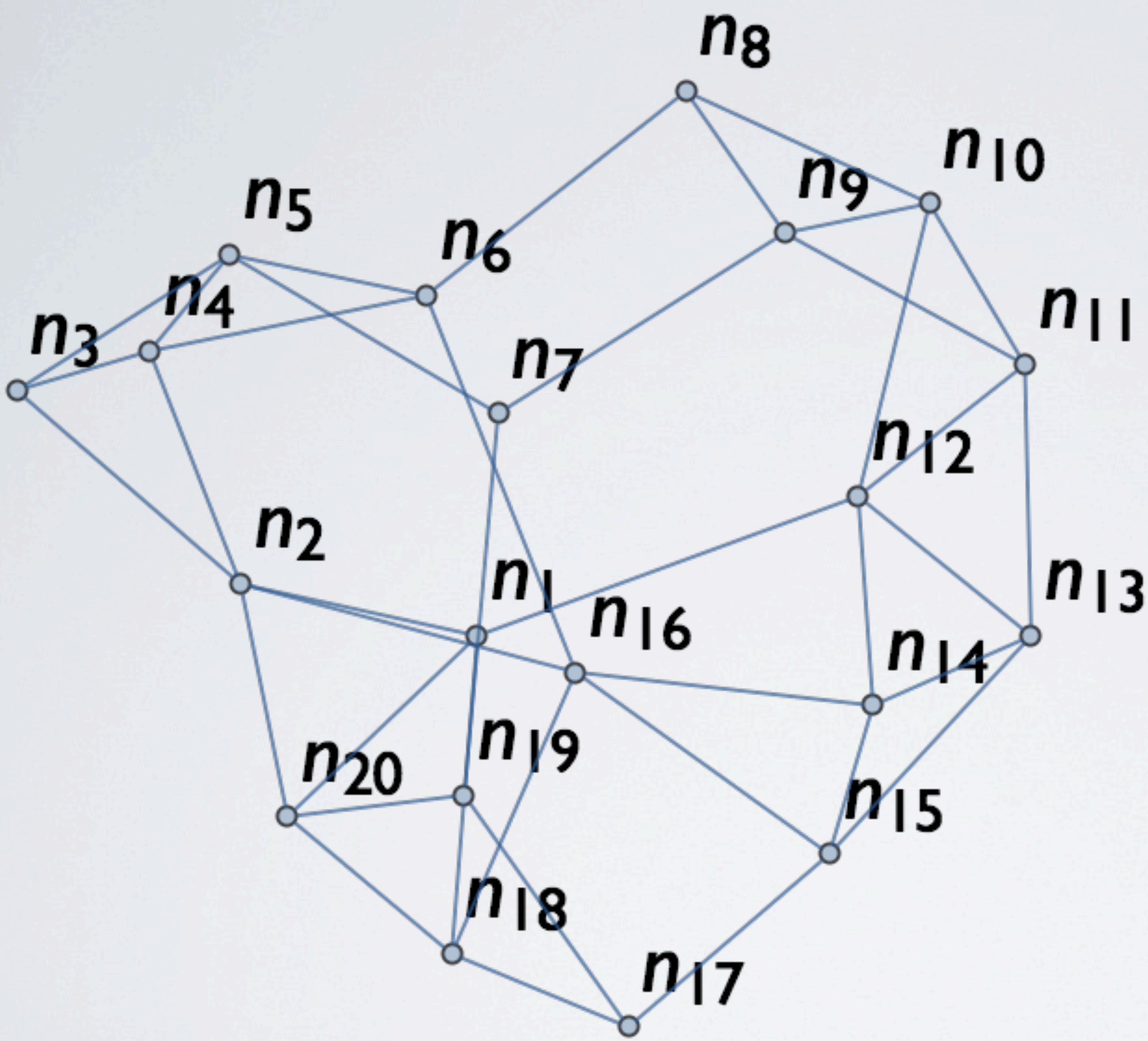
→ write 12

← ack



n_2

→	write	12
←	ack	

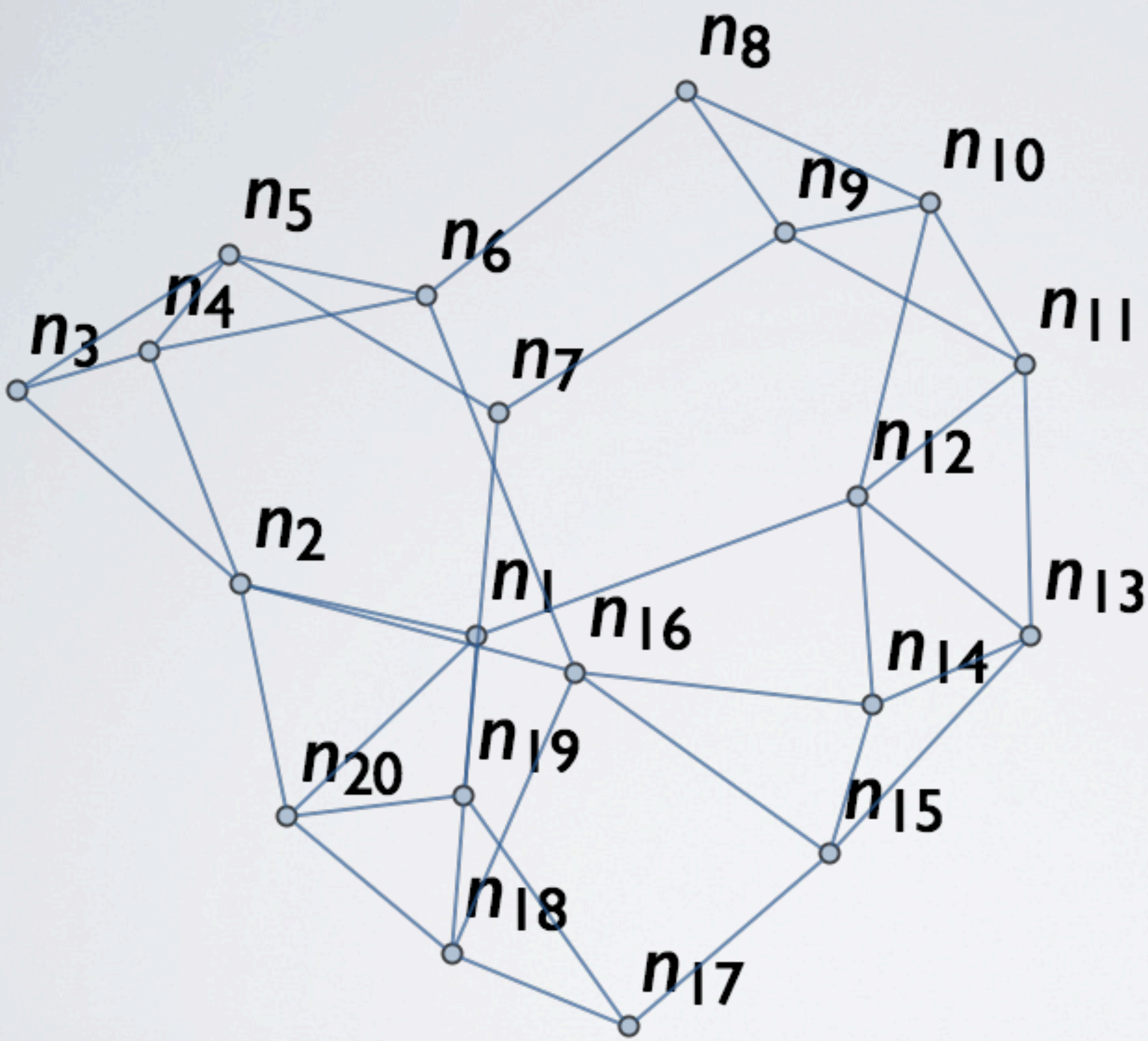


n_2

→ write 12

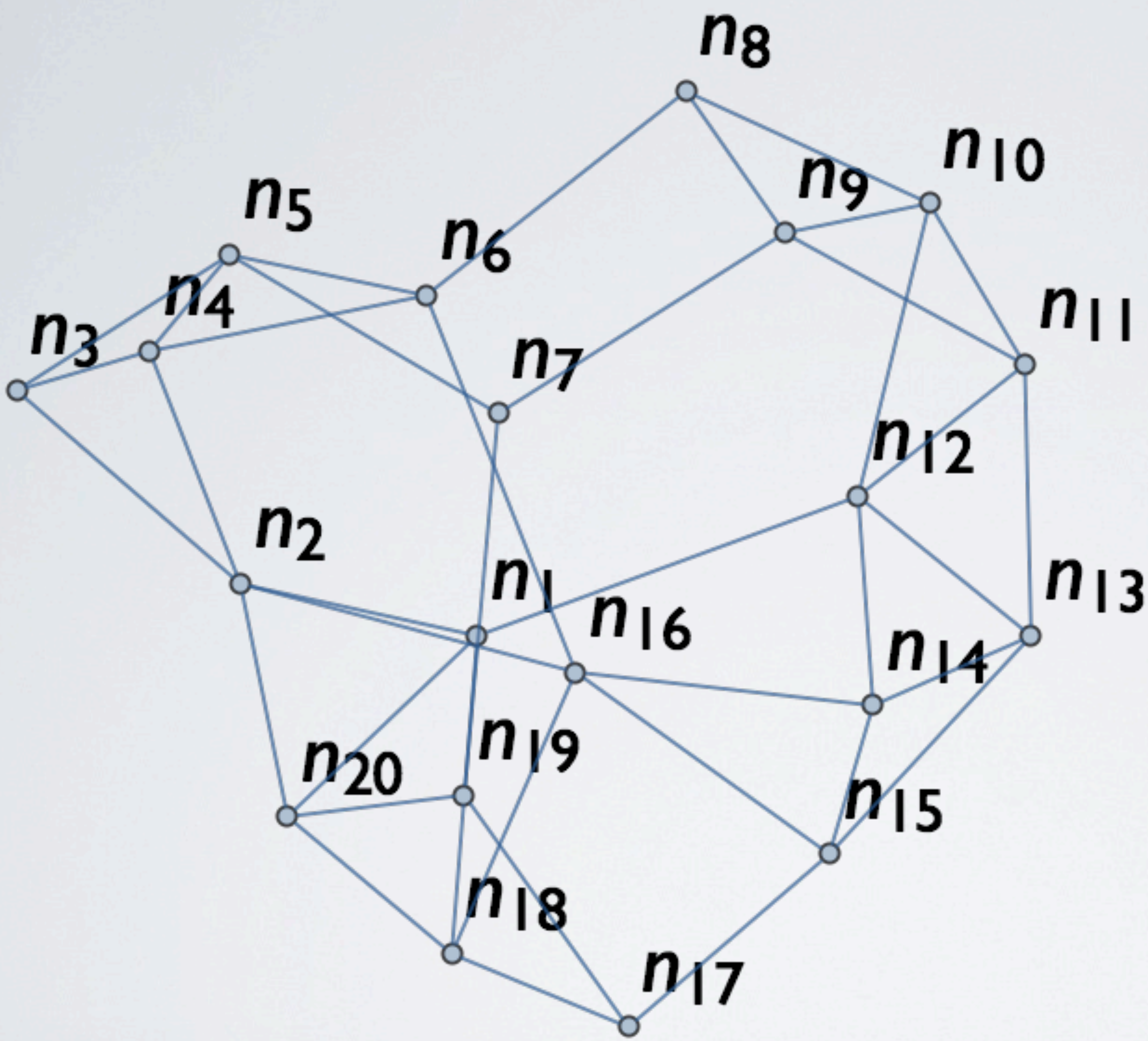
← ack

→ read



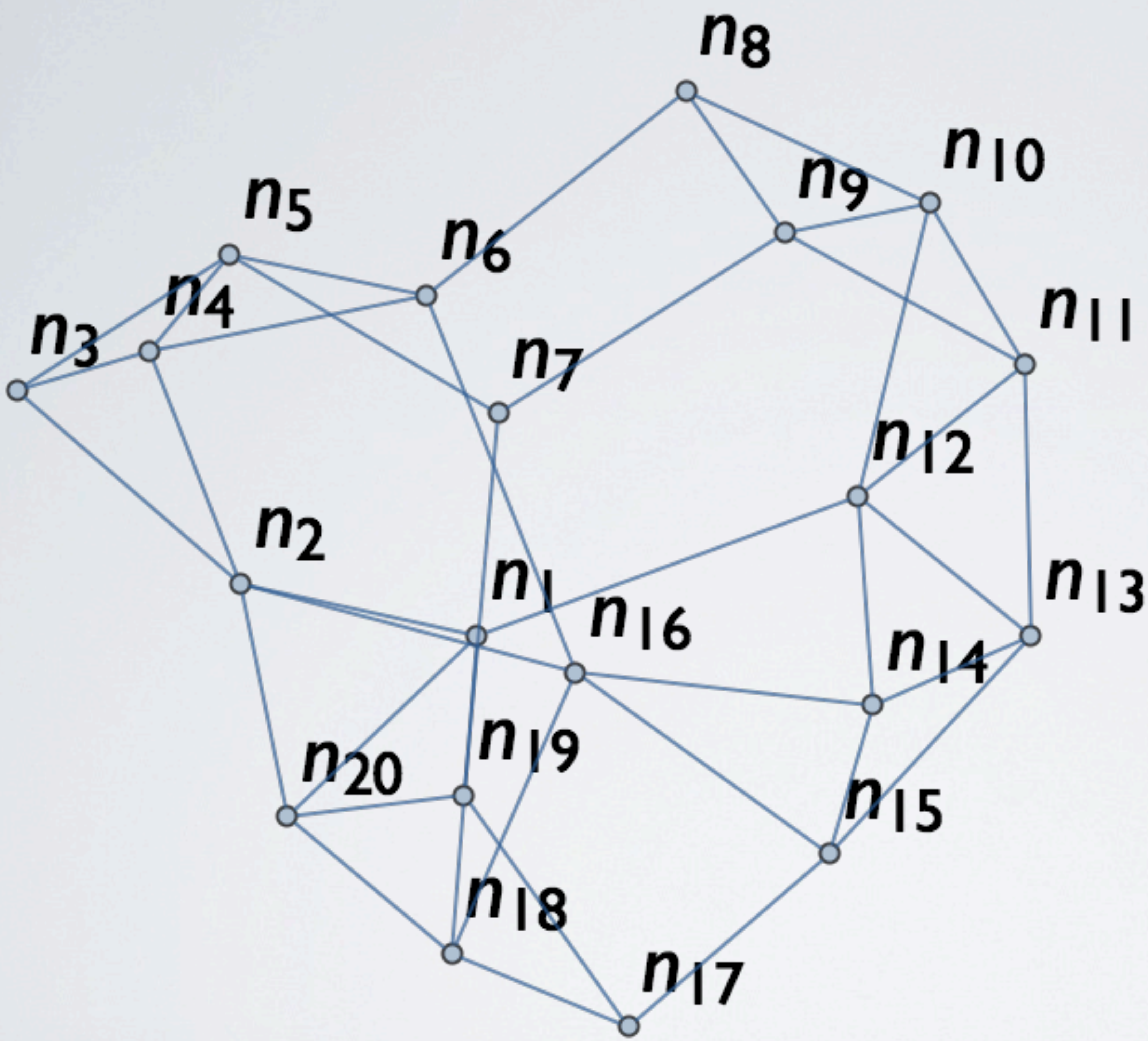
n_2

→	write	12
←	ack	
→	read	
←	12	



n_2

→	write	12
←	ack	
→	read	
←	12	



n_2

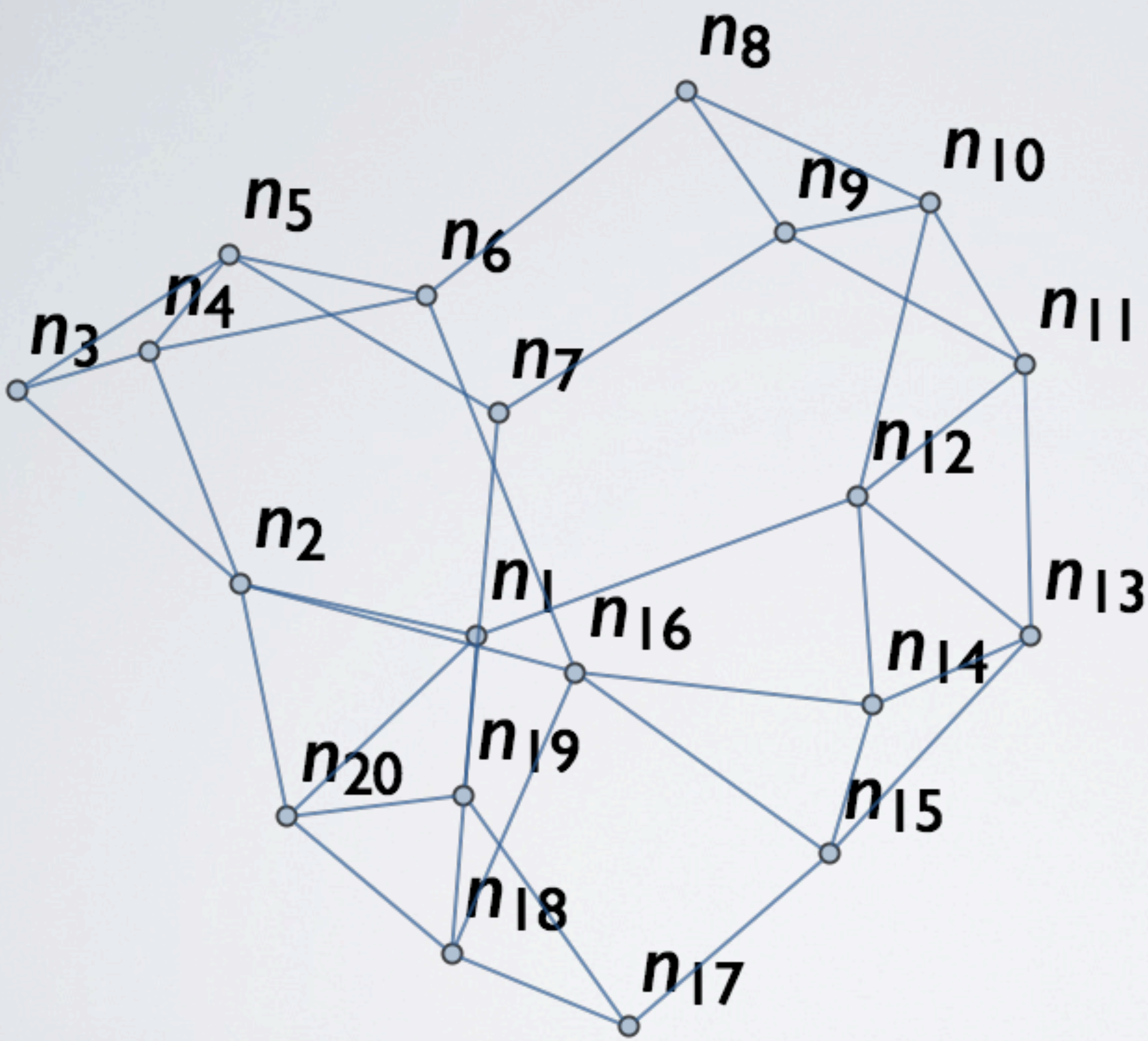
→ write 12

← ack

→ read

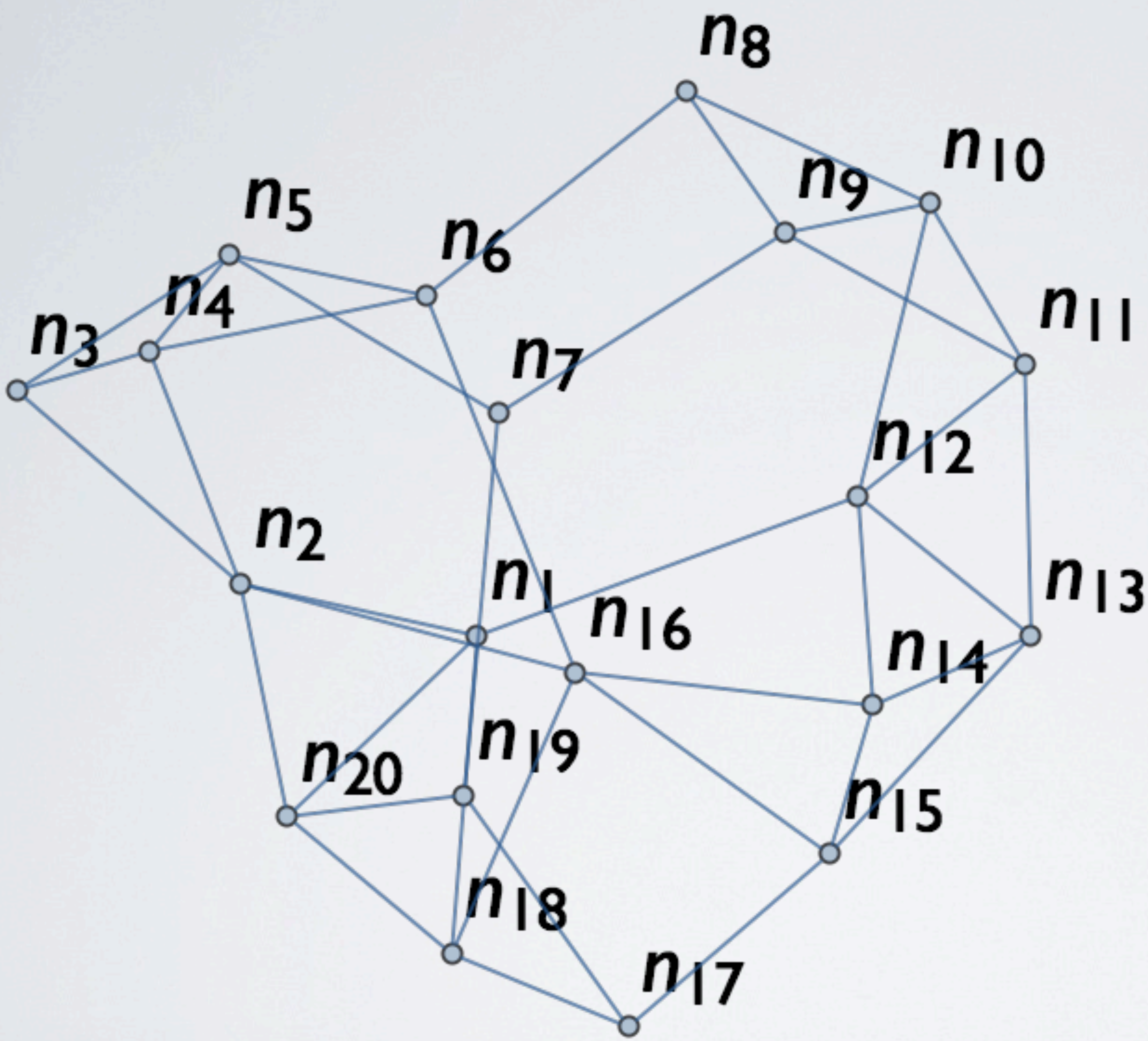
← 12

→ read



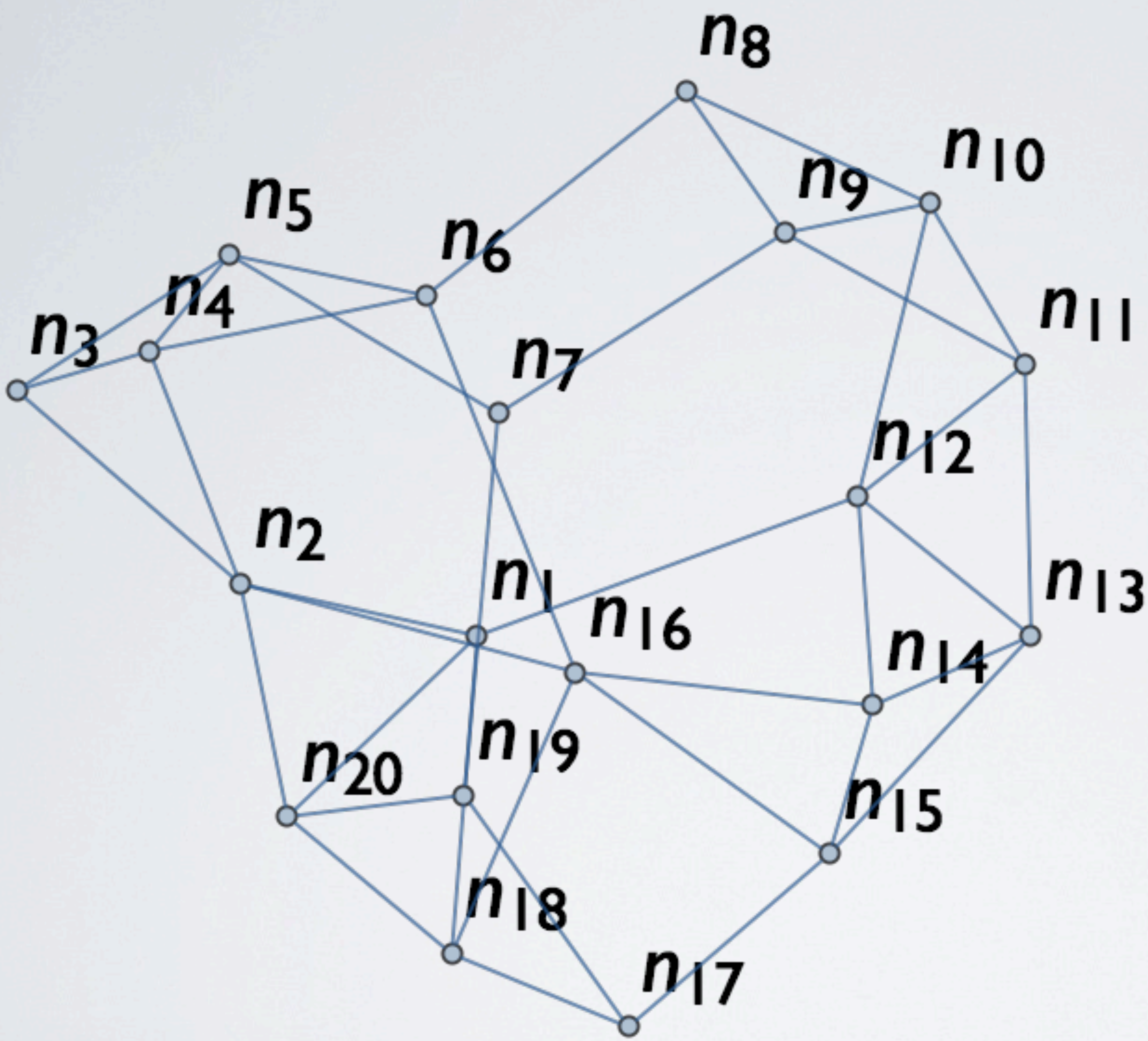
n_2

→	write	12
←	ack	
→	read	
←	12	
→	read	
←	12	



n_2

→	write	12
←	ack	
→	read	
←	12	
→	read	
←	12	



n_2

→ write 12

← ack

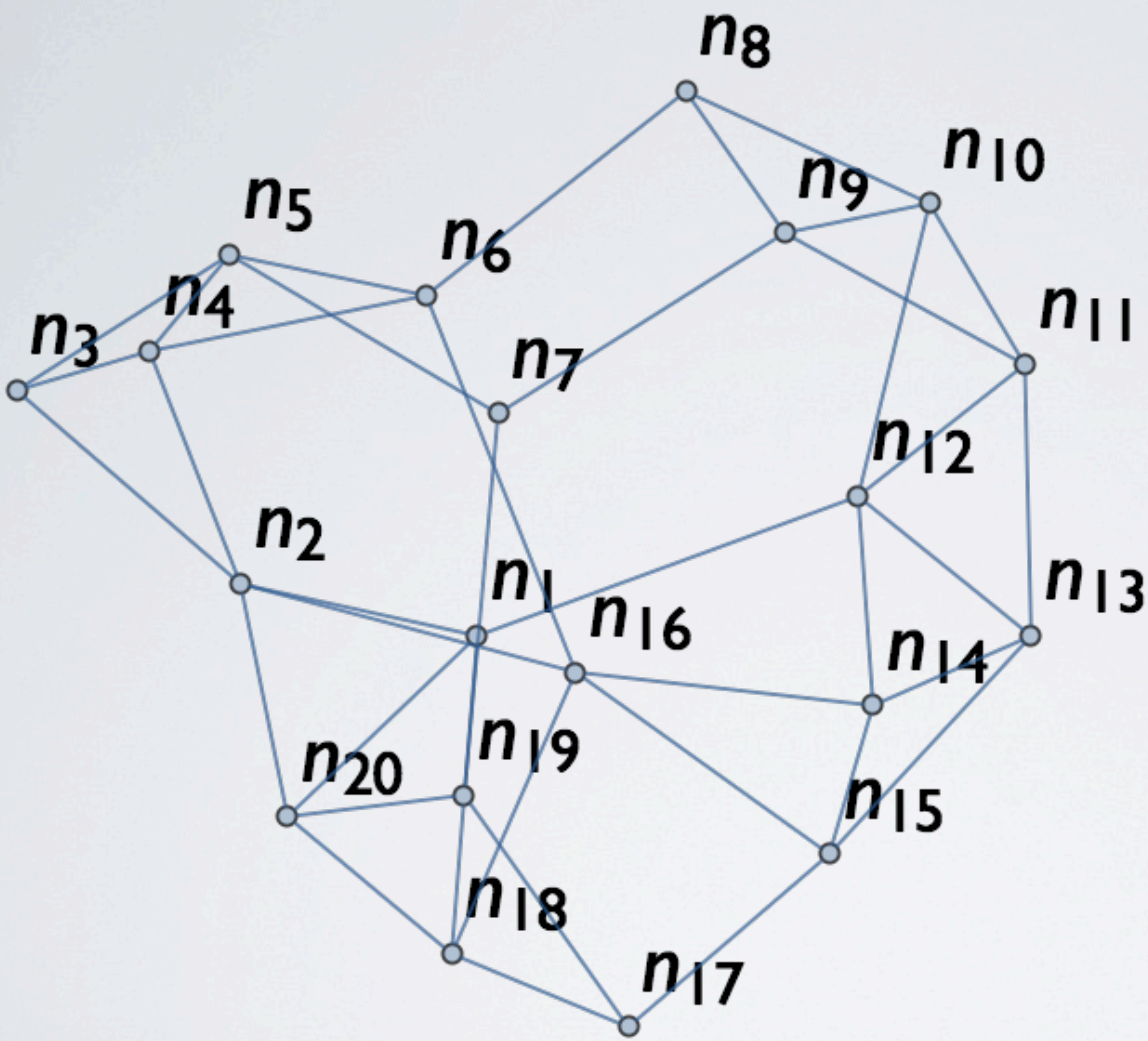
→ read

← 12

→ read

← 12

→ write 20



n_2

→ write 12

← ack

→ read

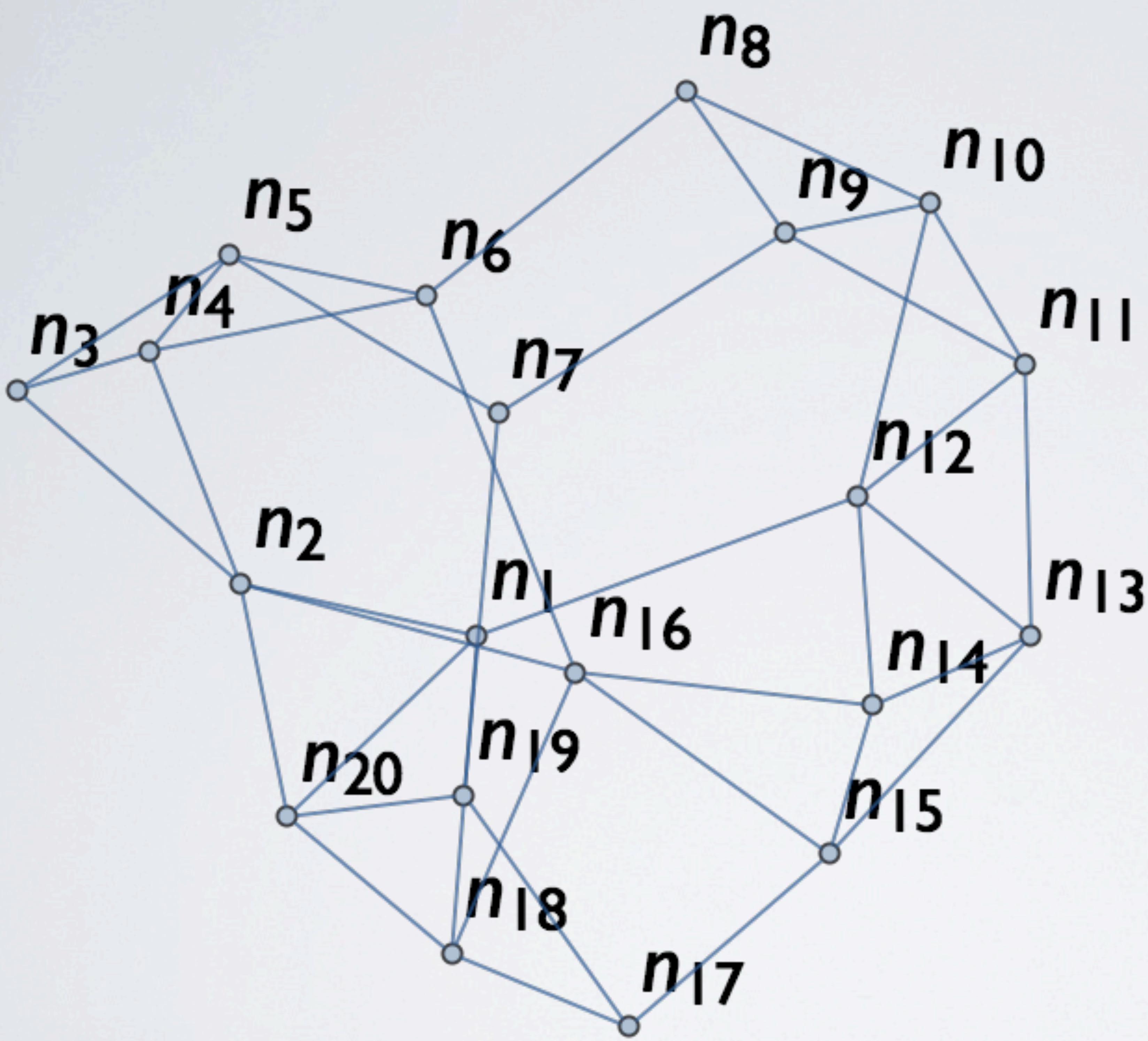
← 12

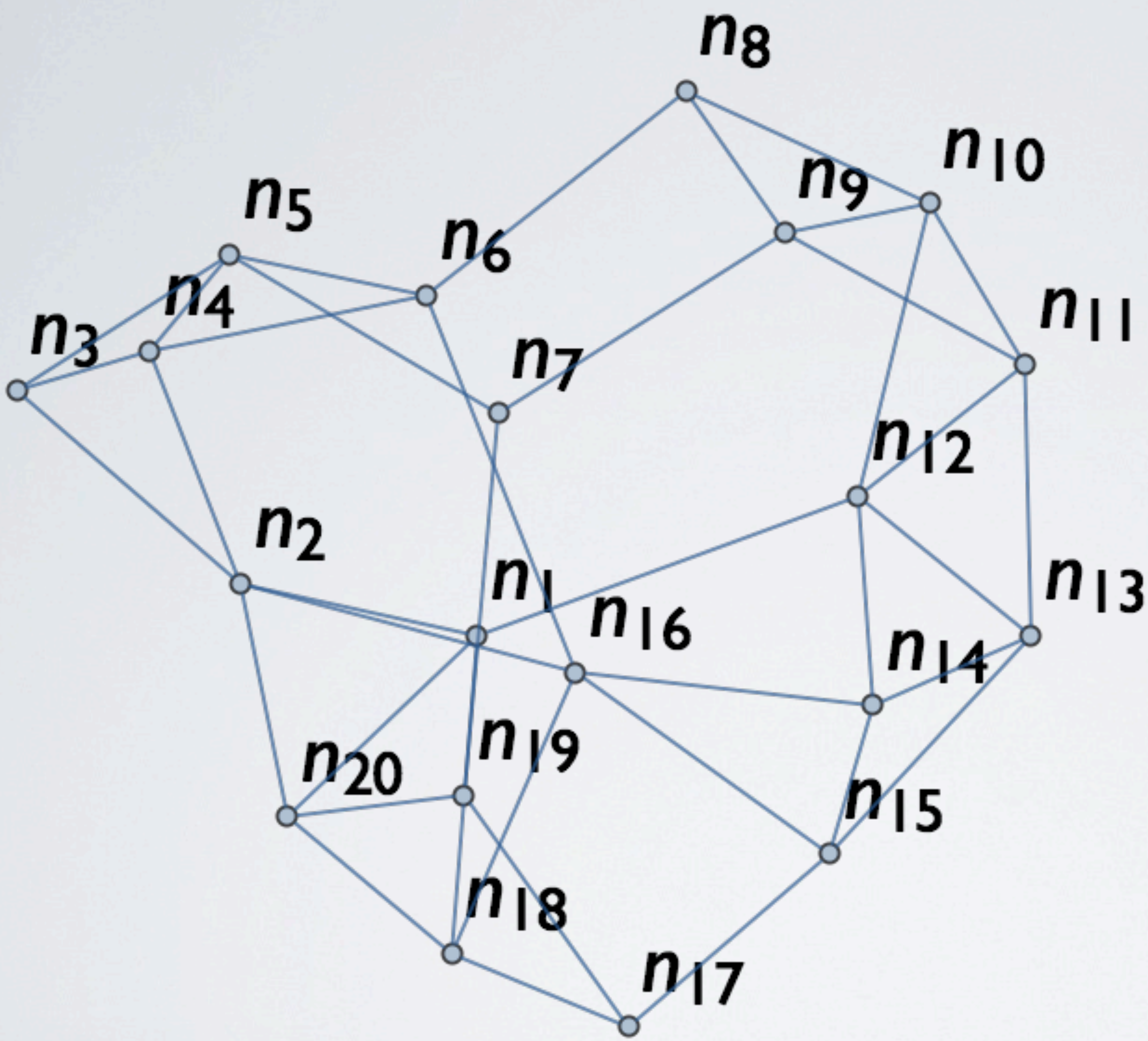
→ read

← 12

→ write 20

← ack


n_8
n_{10}
n_9
n_5
n_6
n_4
n_{11}
n_3
n_7
n_{12}
n_2
n_{13}
n_1
n_{16}
n_{14}
n_{19}
n_{20}
n_{15}
n_{18}
n_{17}



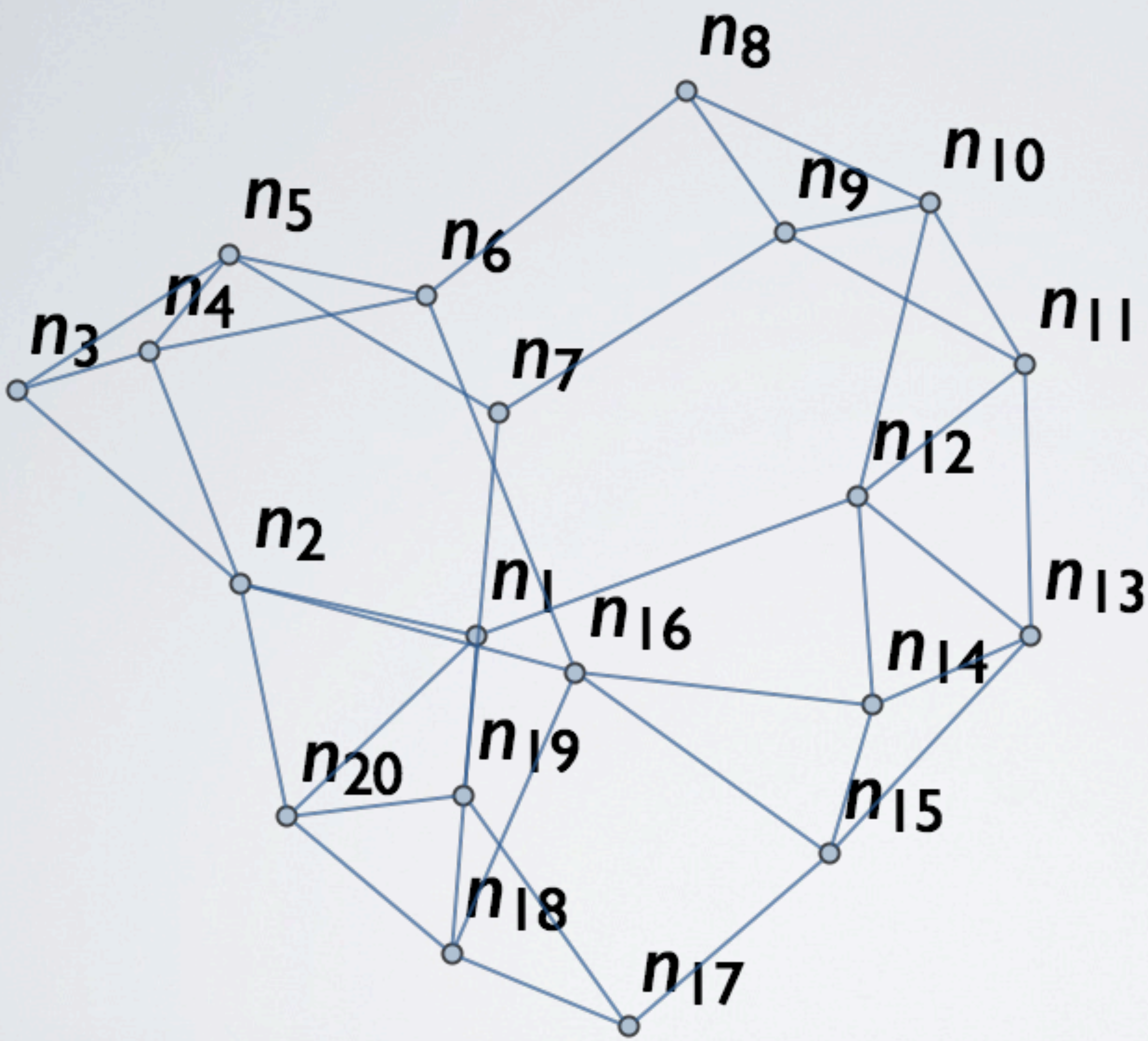
n_2

write 12

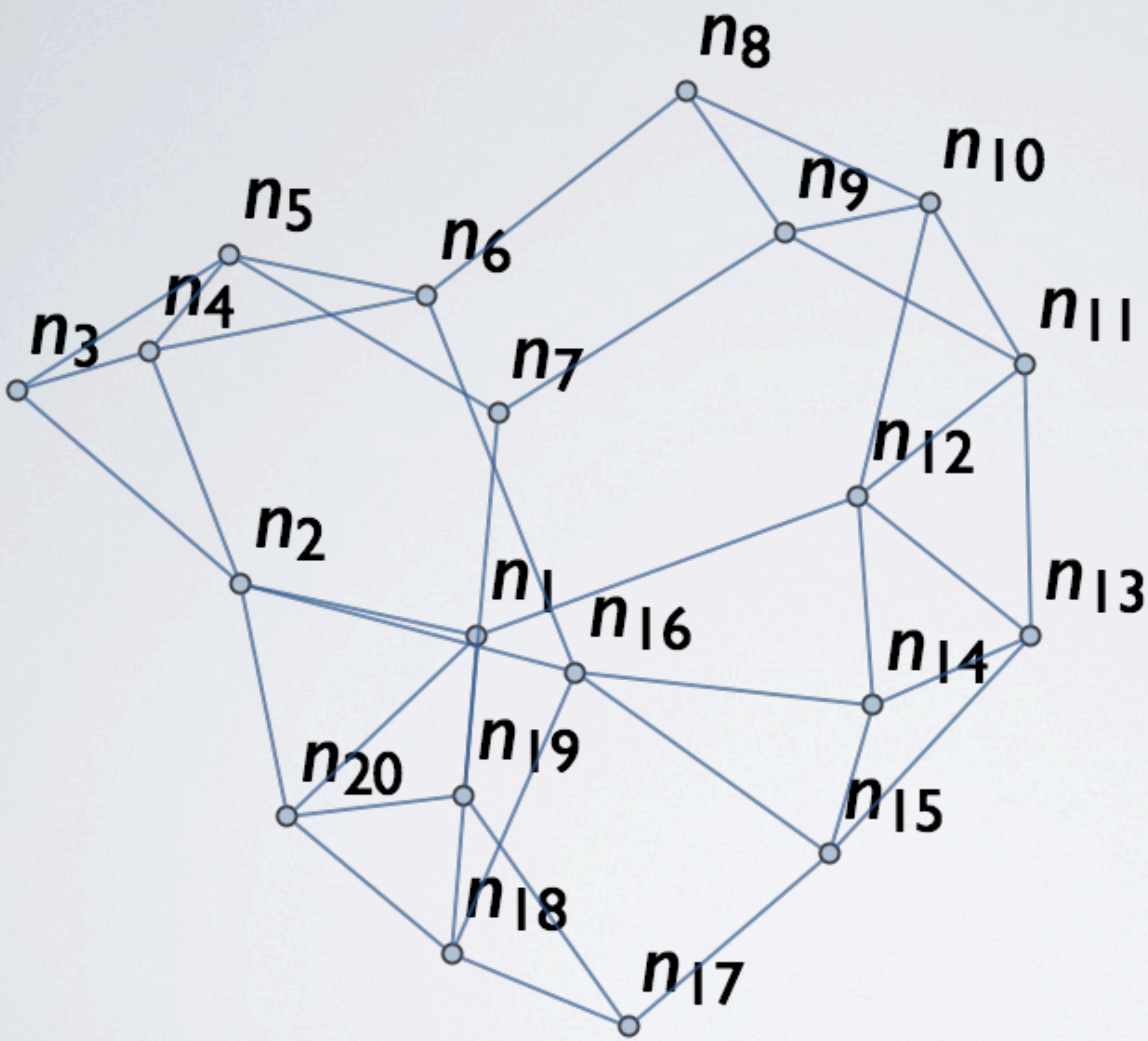
read

read

write 20

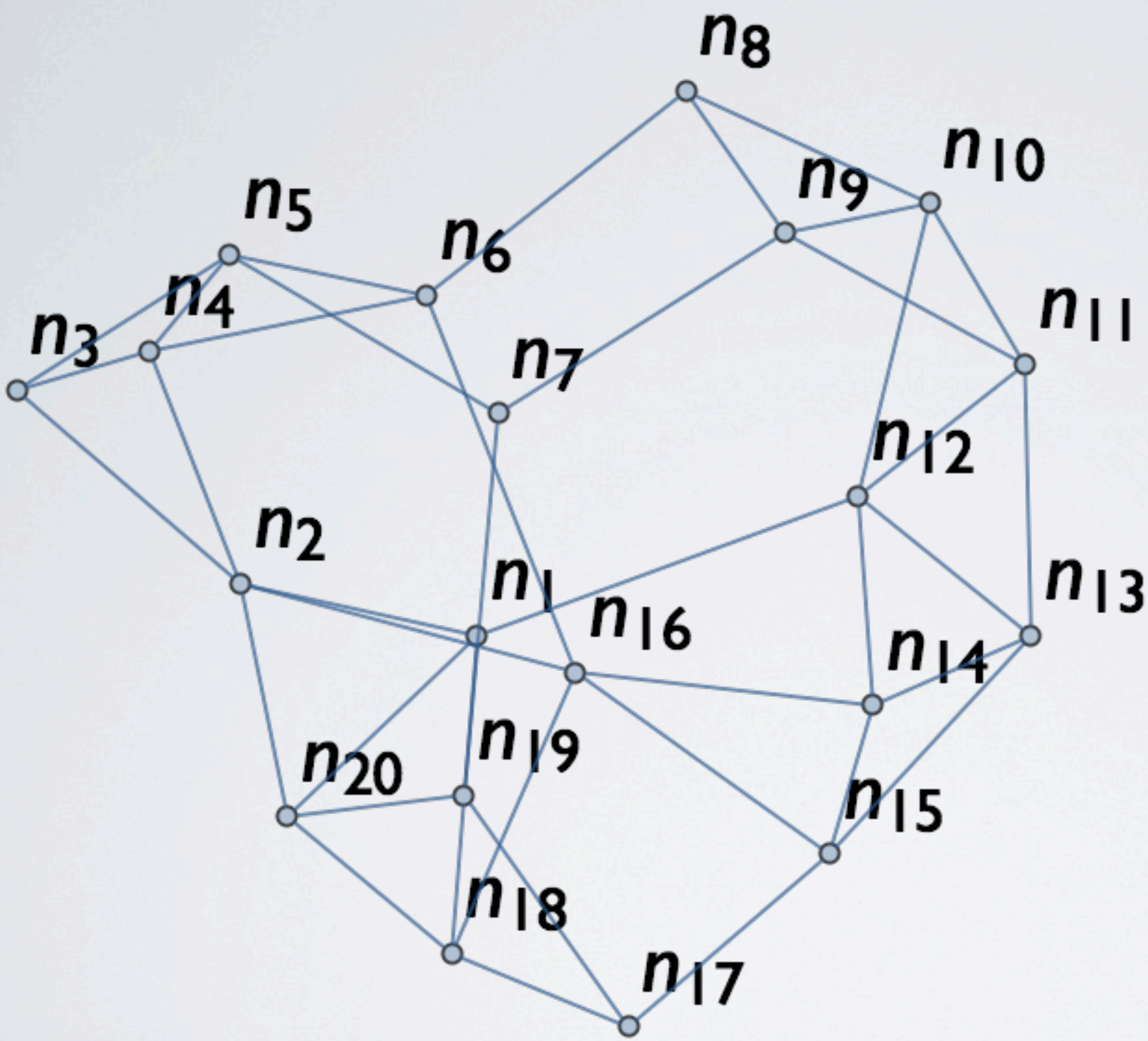


n_2	n_{11}
write 12	write 12
read	read
read	read
write 20	write 20



n_2	n_{11}
write 12	write 12
read	read
read	read
write 20	write 20

n_{17}
write 12
write 9
read
write 20



$\boldsymbol{n_2}$	$\boldsymbol{n_{11}}$
write 12	write 12
read	read
read	read
write 20	write 20

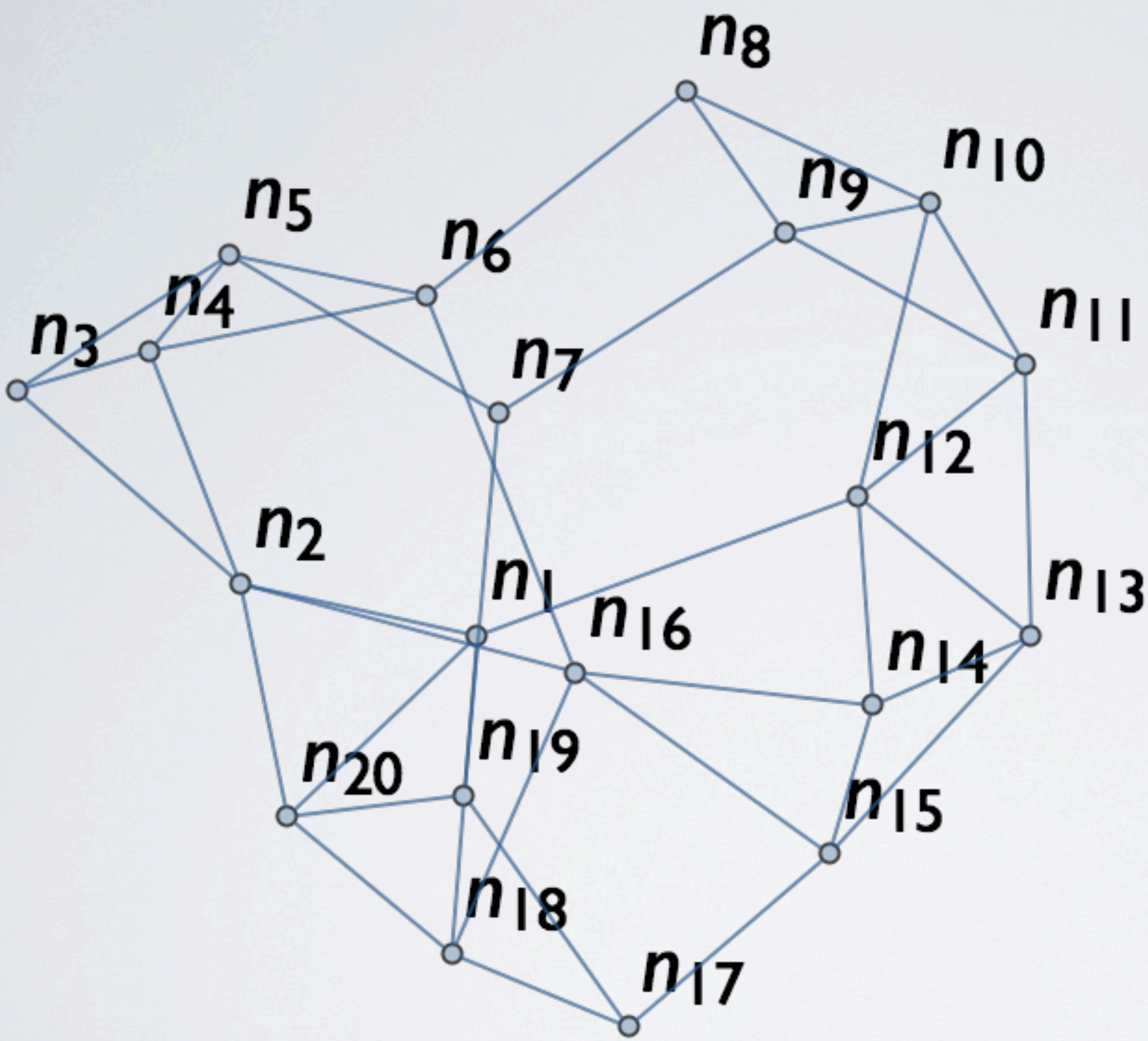
$\boldsymbol{n_{17}}$

write 12

write 9

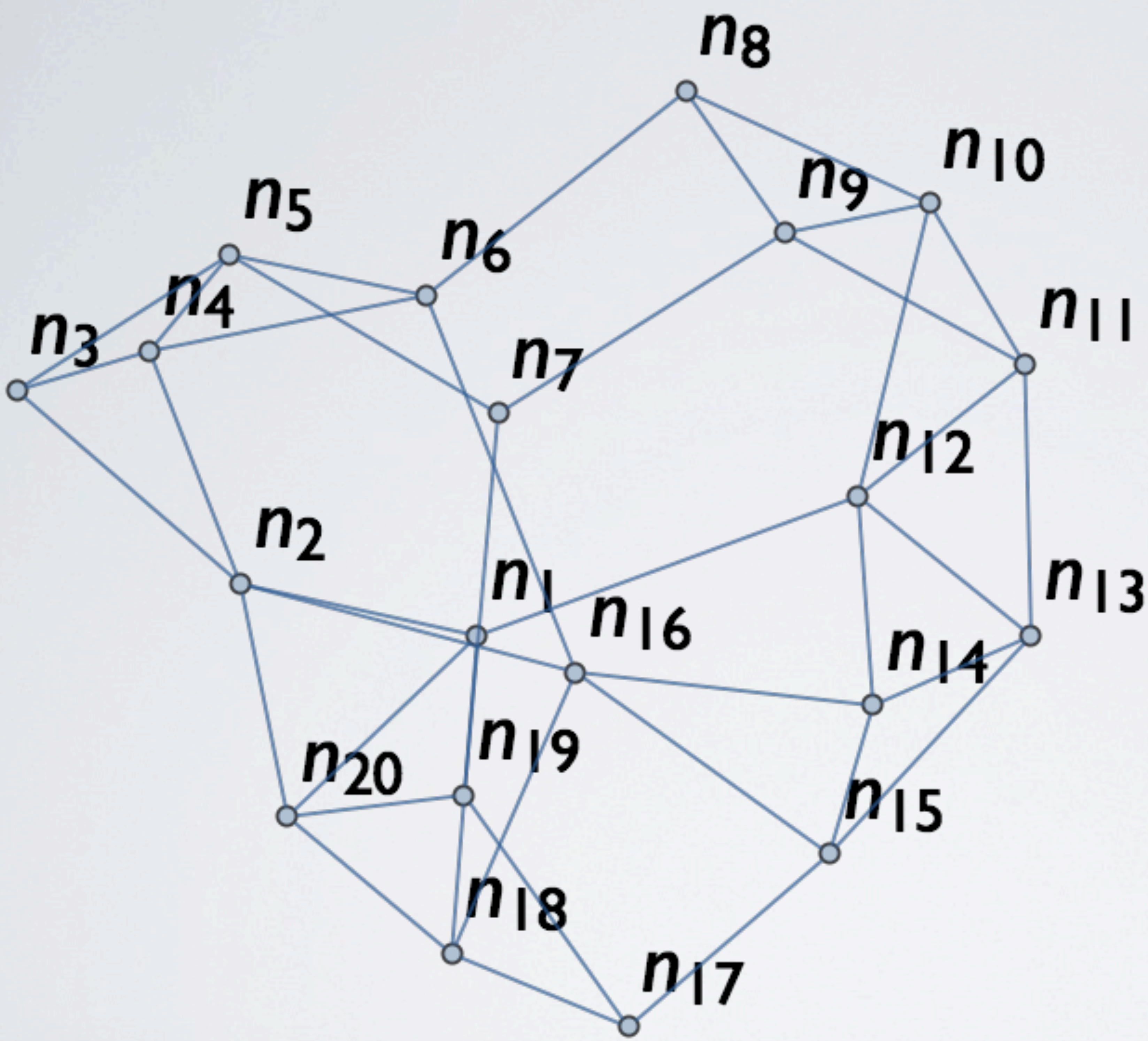
read

write 20

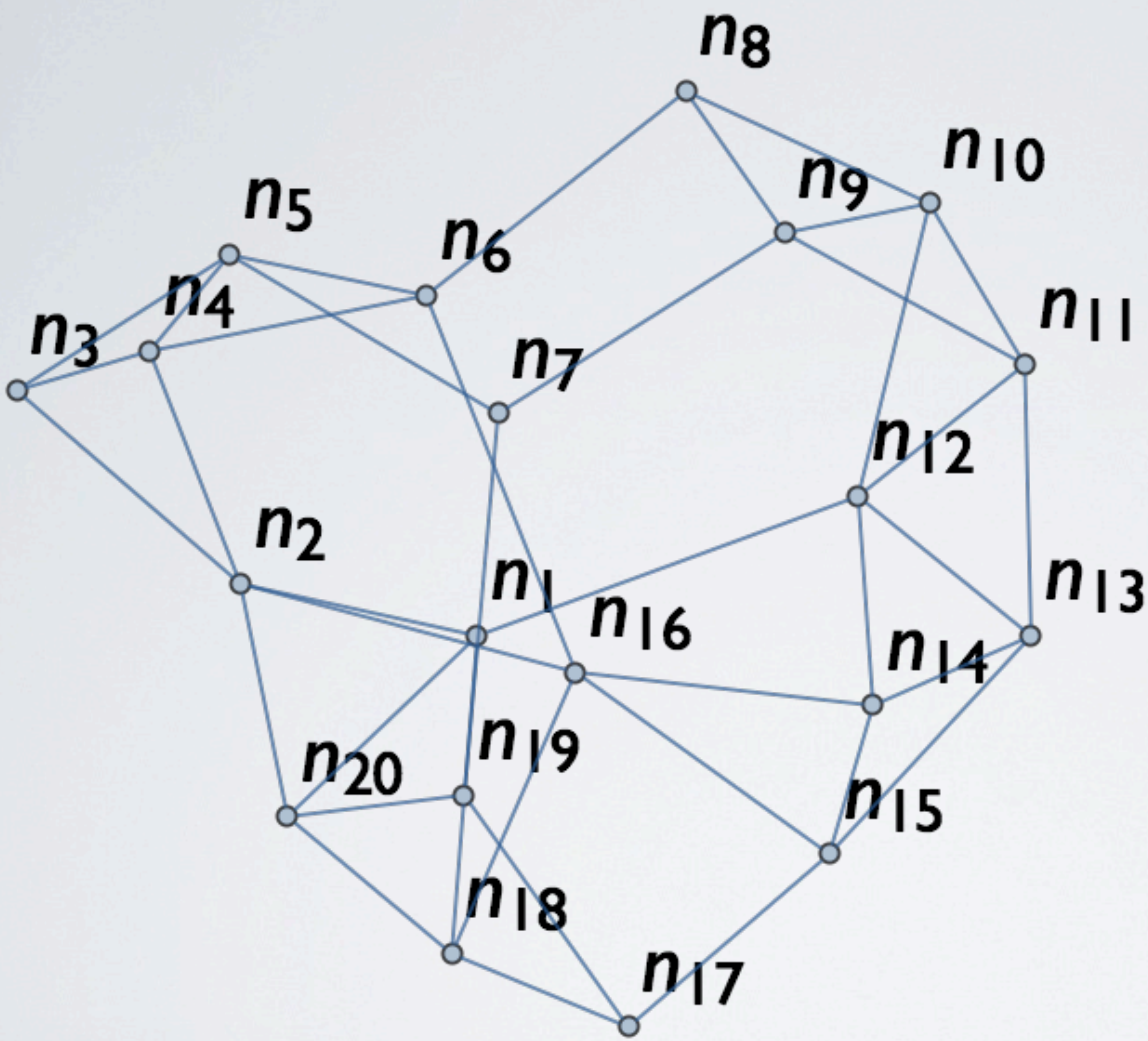


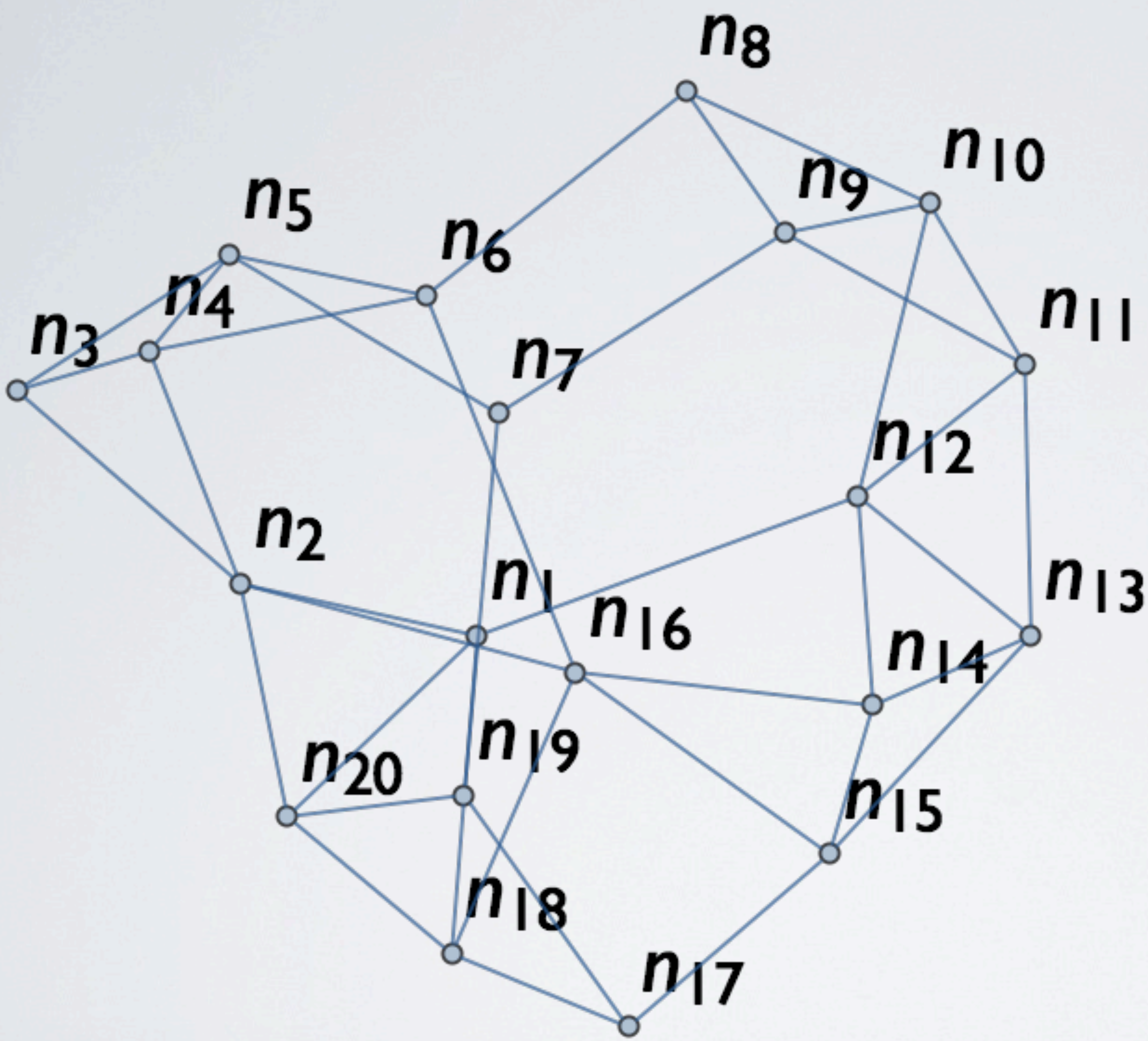
Atomic
Linearizable

Availability


n_8
n_{10}
n_9
n_5
n_6
n_4
n_{11}
n_3
n_7
n_{12}
n_2
n_{13}
n_1
n_{16}
n_{14}
n_{19}
n_{20}
n_{15}
n_{18}
n_{17}

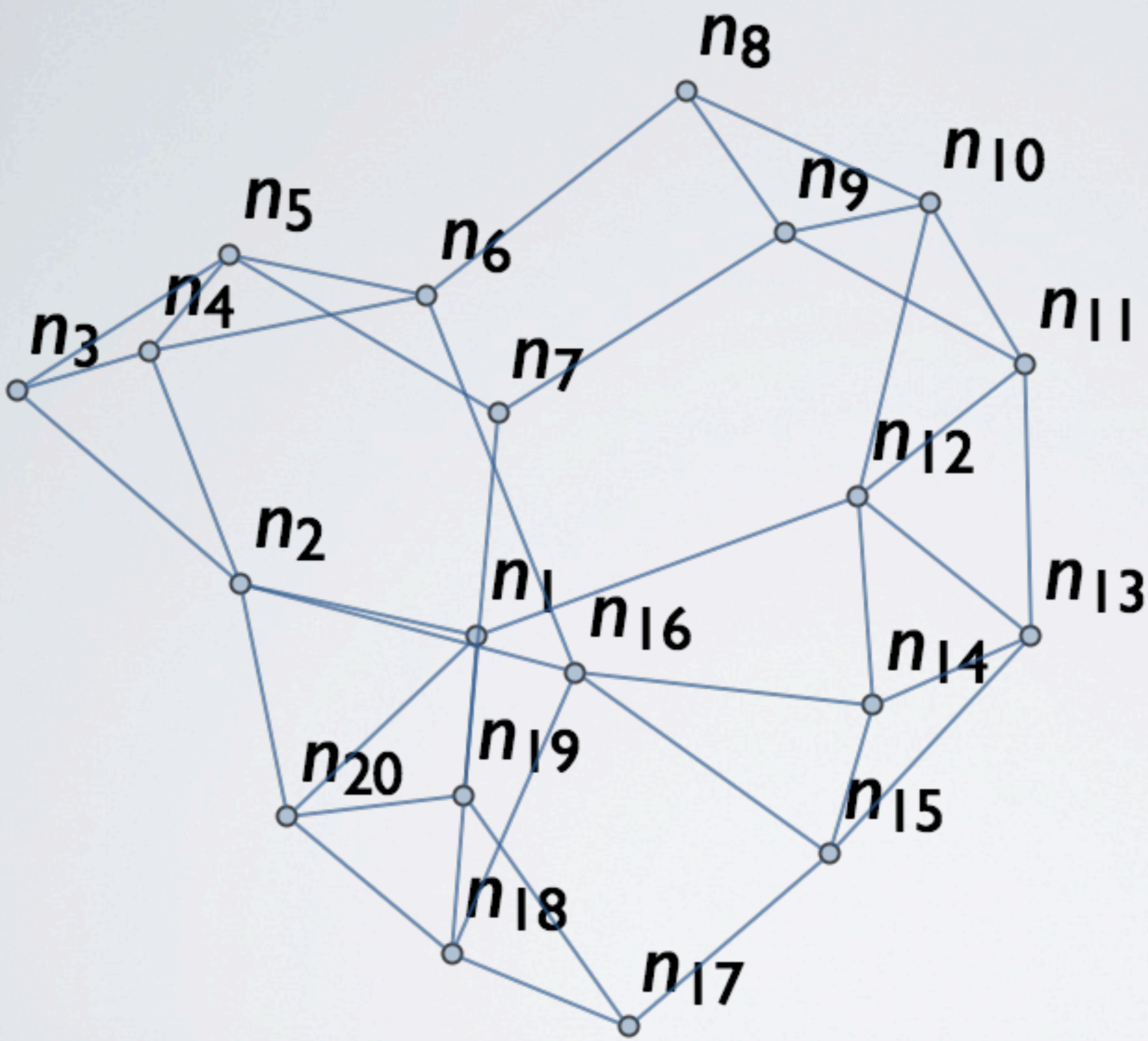
n_2


n8
n10
n9
n5
n6
n4
n11
n3
n7
n12
n2
n13
n1
n16
n14
n19
n20
n15
n18
n17


n_8
n_9
n_{10}
n_5
n_6
n_4
n_3
n_7
n_{11}
n_{12}
n_2
n_1
n_{16}
n_{13}
n_{14}
n_{20}
n_{19}
n_{15}
n_{18}
n_{17}

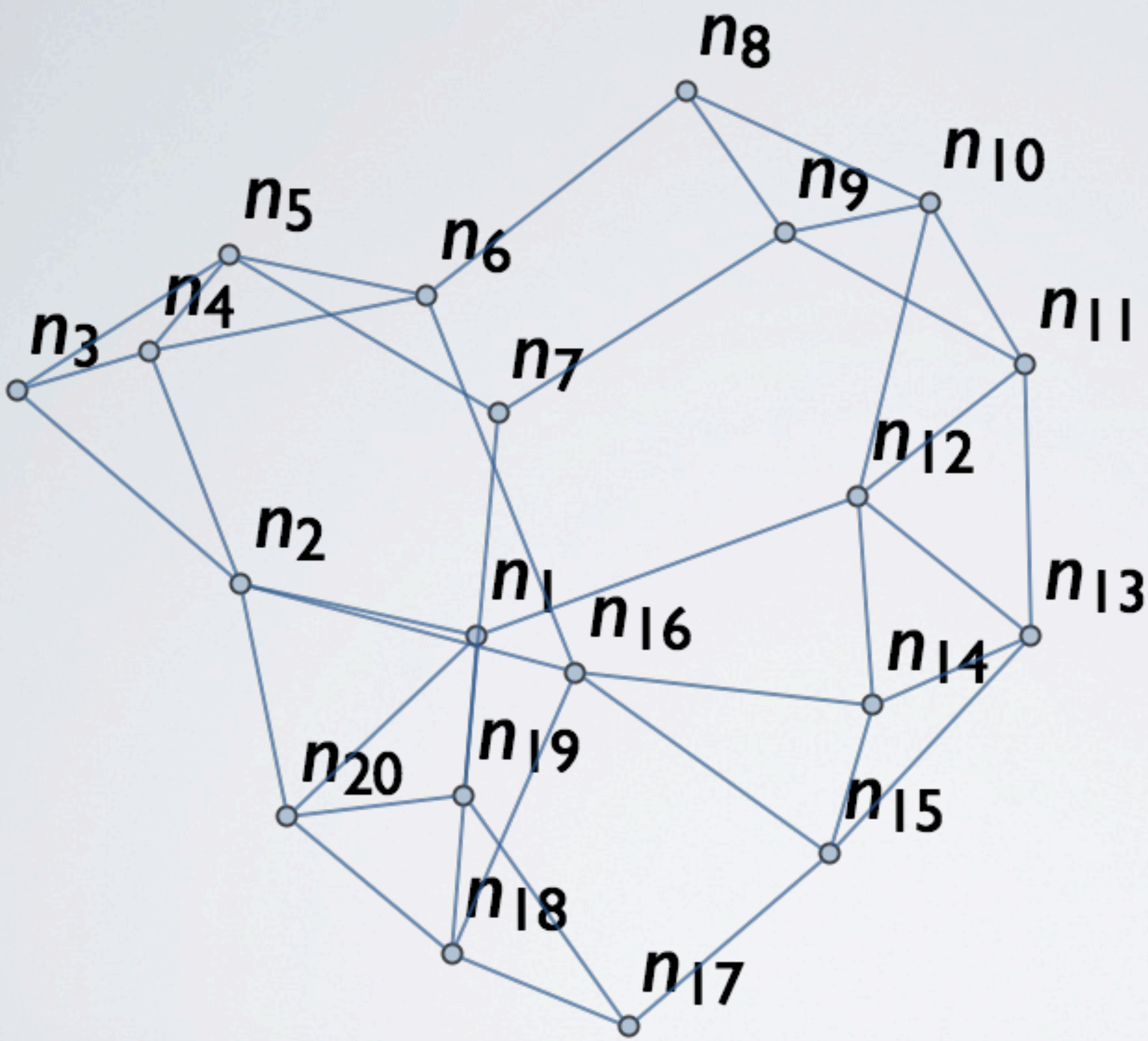
n_2

→ write 12



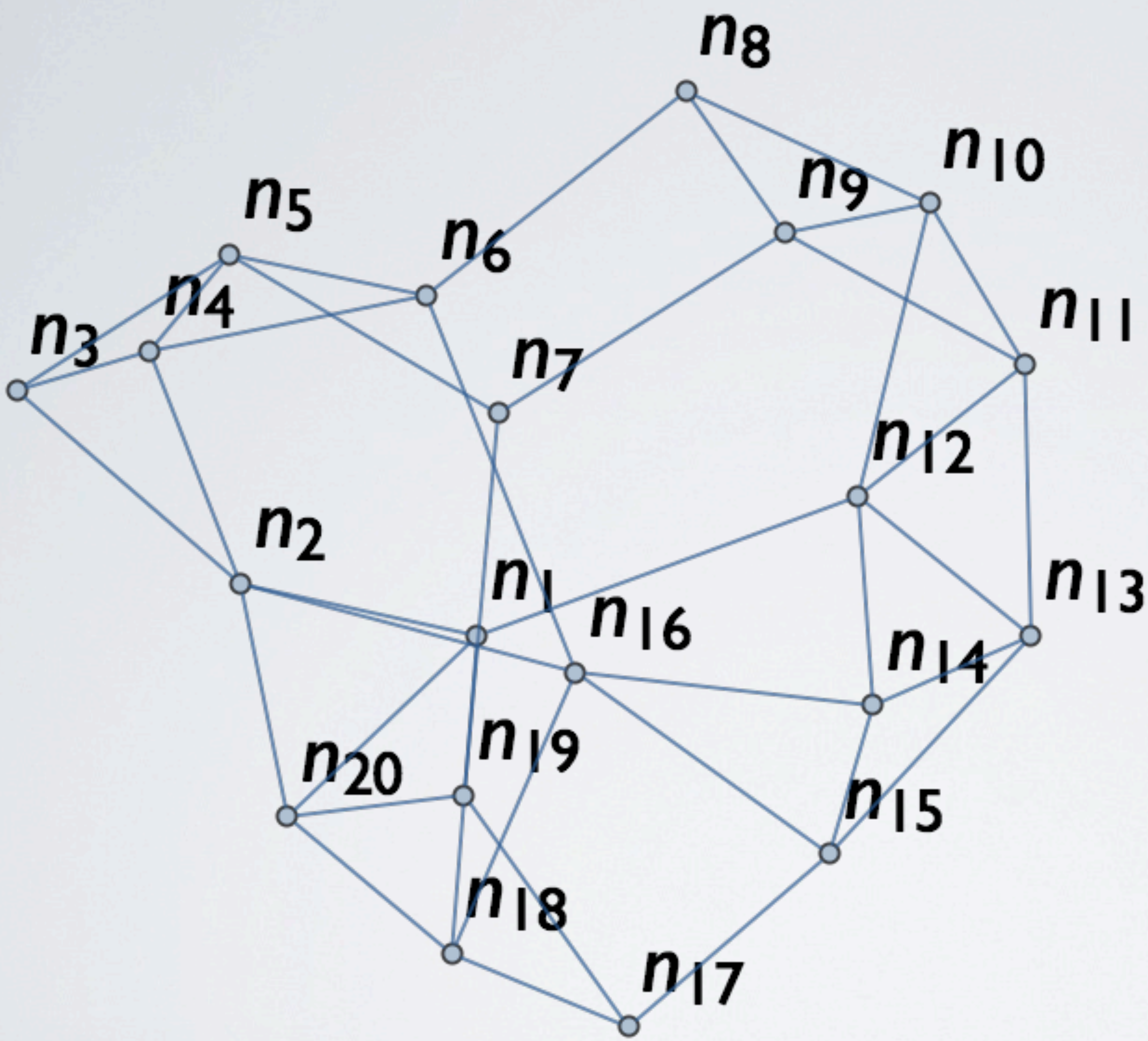
n_2

→	write	12
←	ack	



n_2

→	write	12
←	ack	

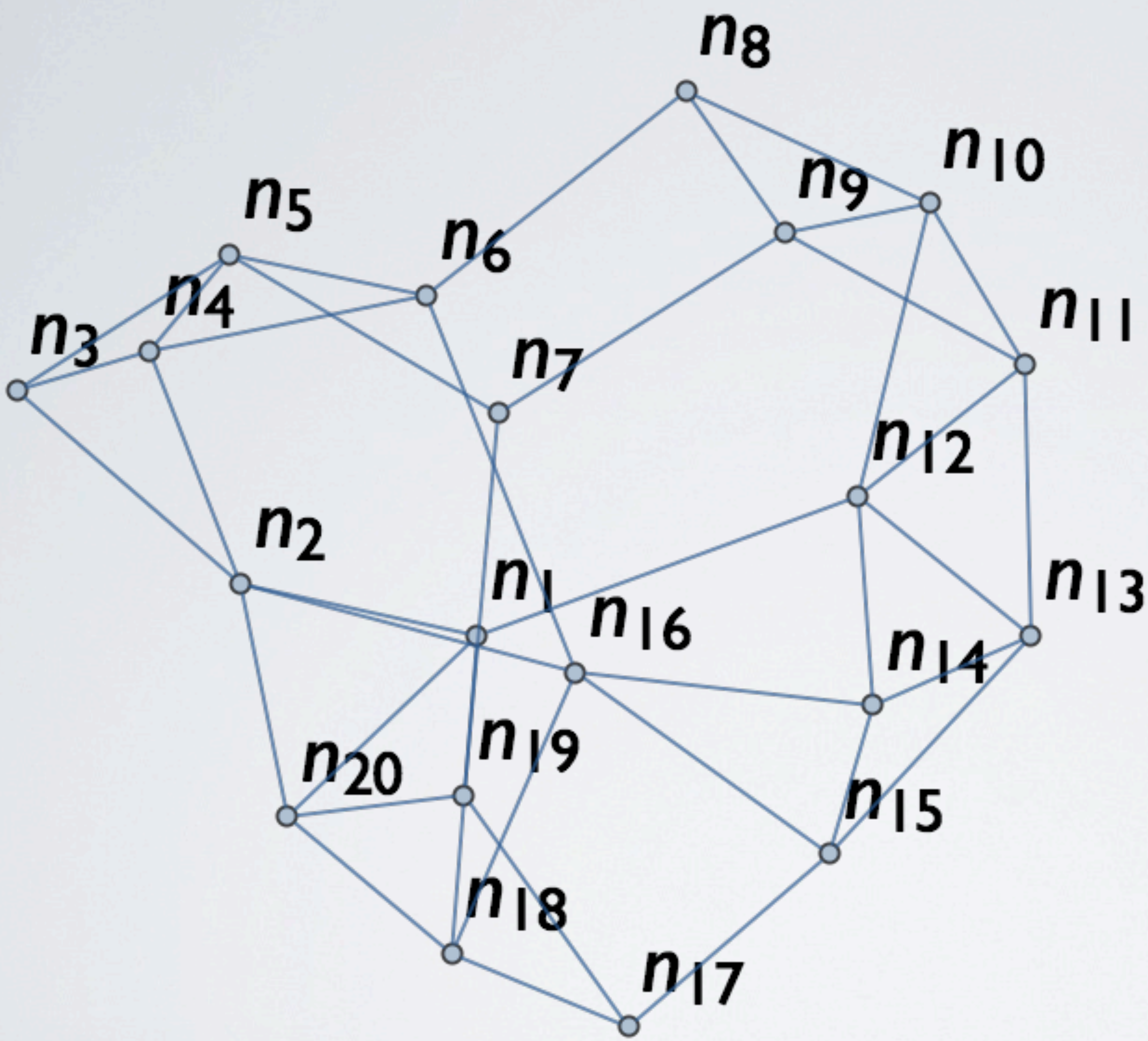


n_2

→ write 12

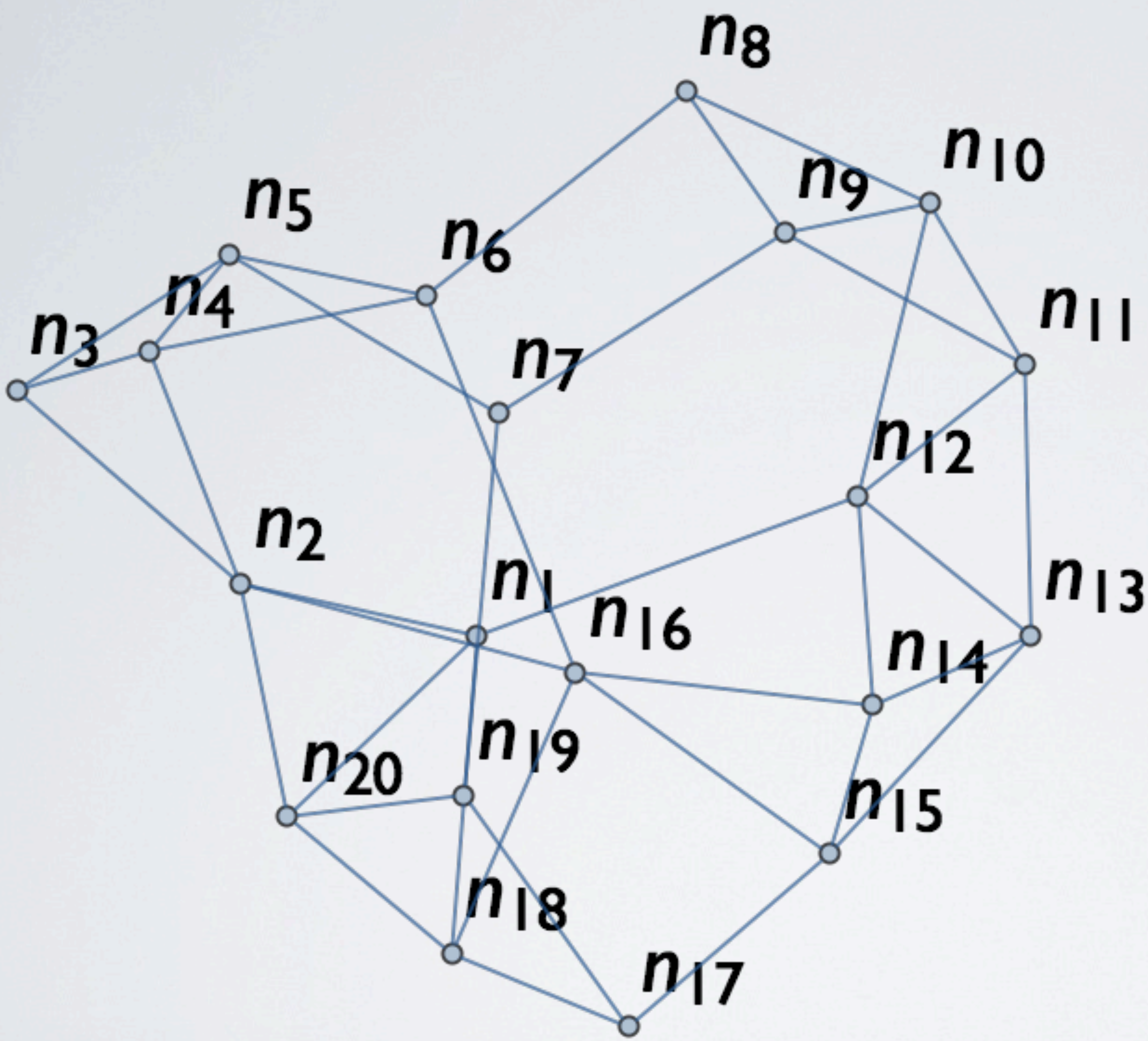
← ack

→ read



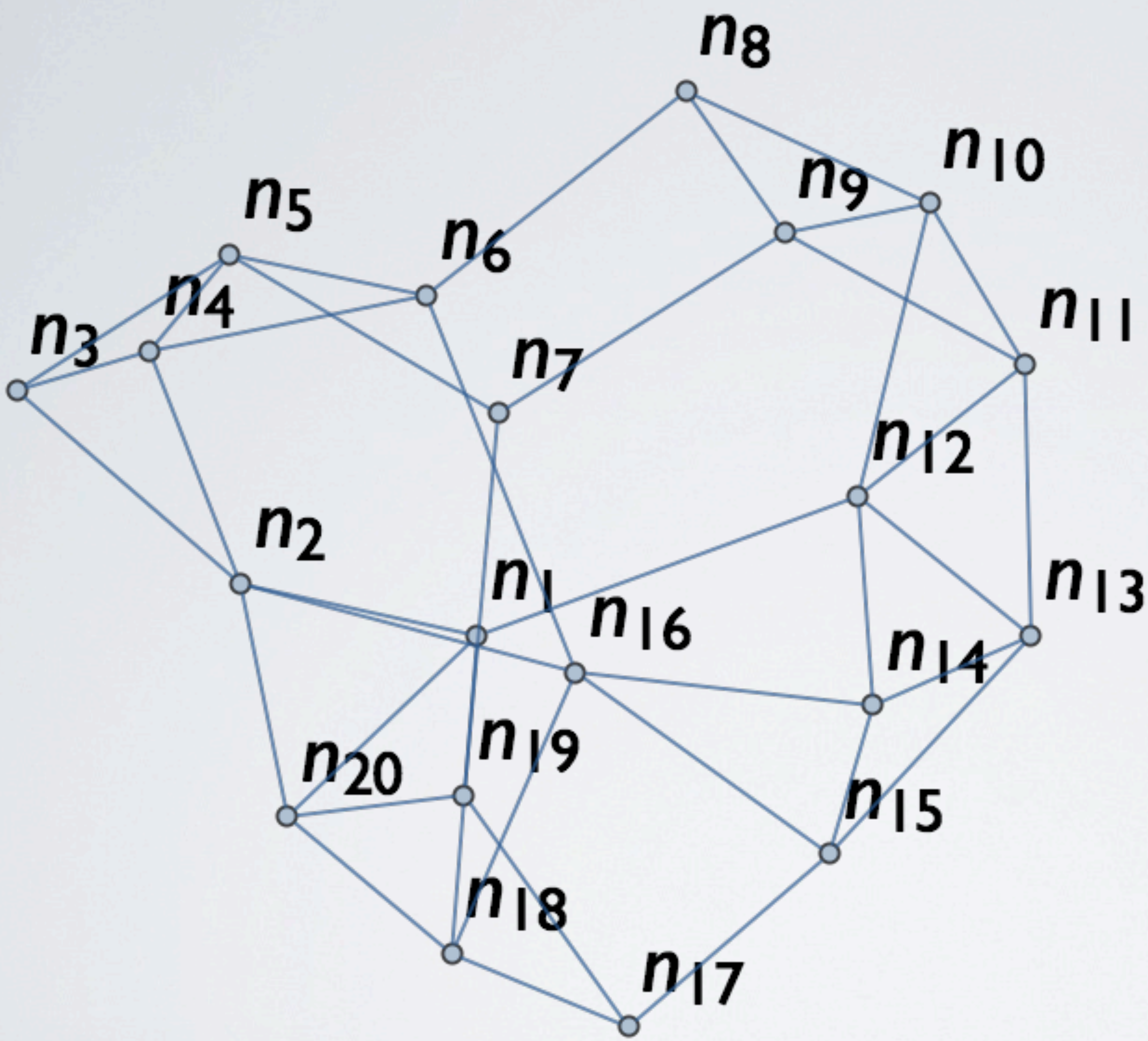
$\mathbf{n_2}$

→	write	12
←	ack	
→	read	
←	12	



n_2

→	write	12
←	ack	
→	read	
←	12	



n_2

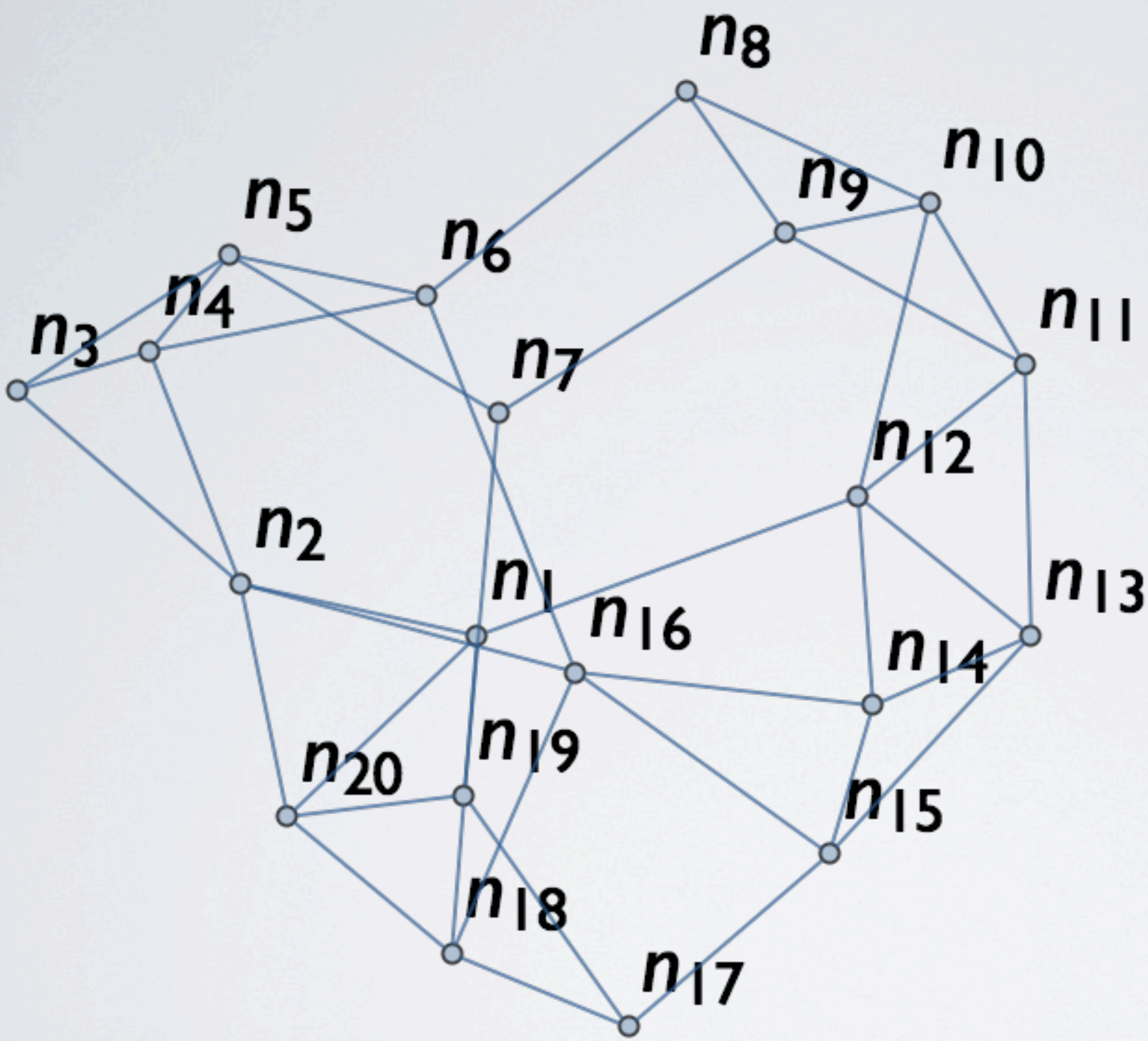
→ write 12

← ack

→ read

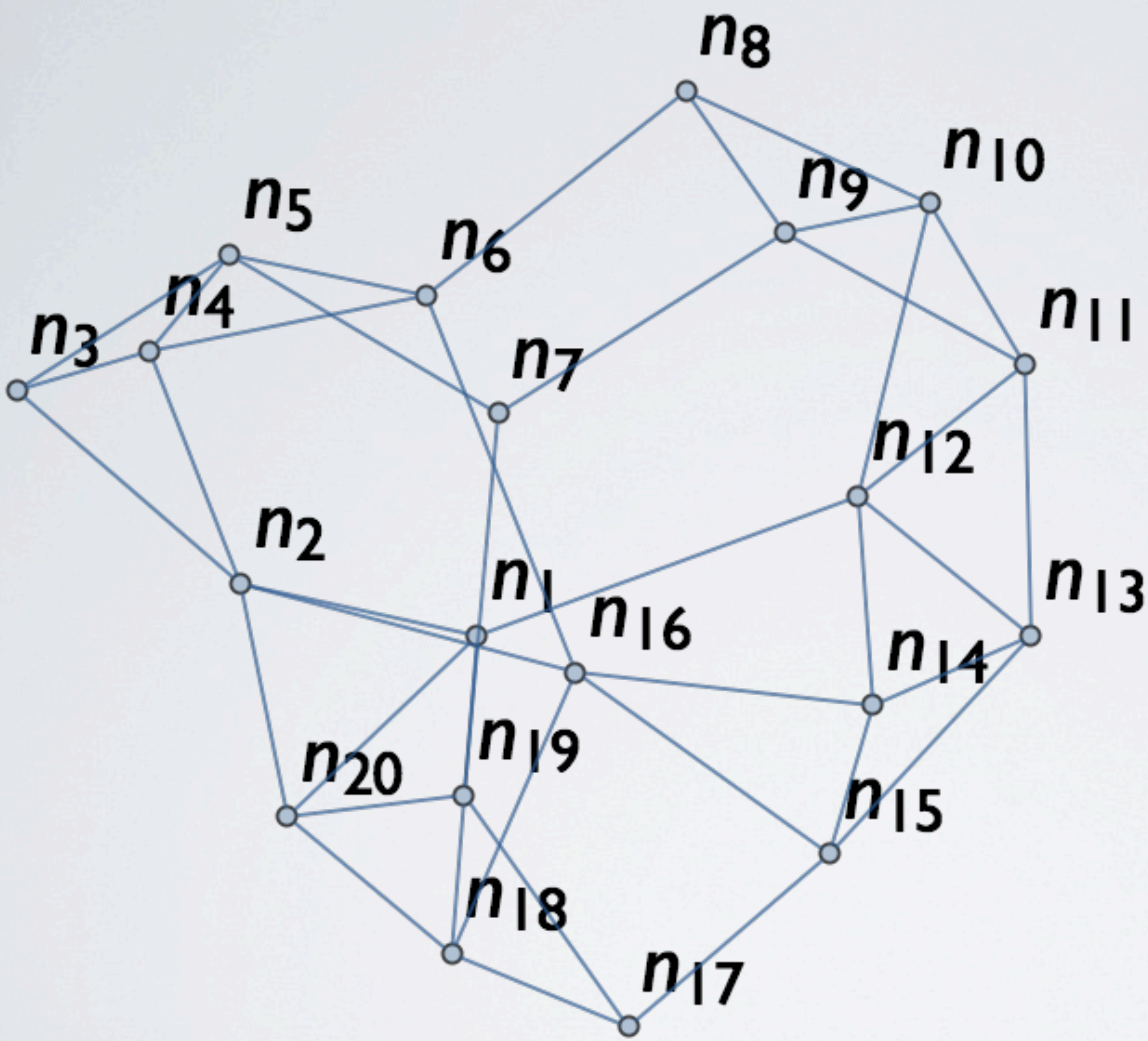
← 12

→ read



n_2

→	write	12
←	ack	
→	read	
←	12	
→	read	



n_2

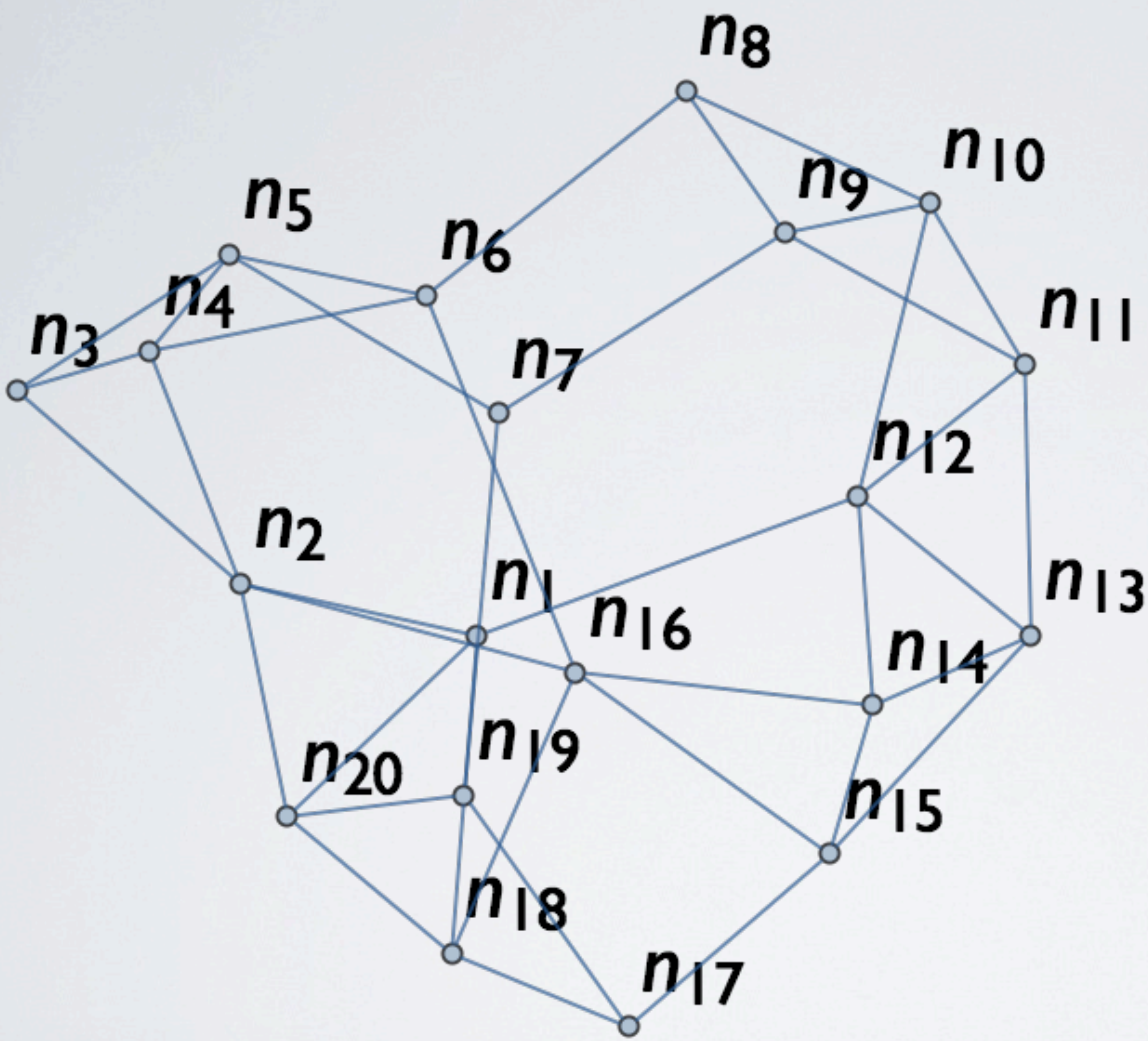
→ write 12

← ack

→ read

← 12

→ read



n_2

→ write 12

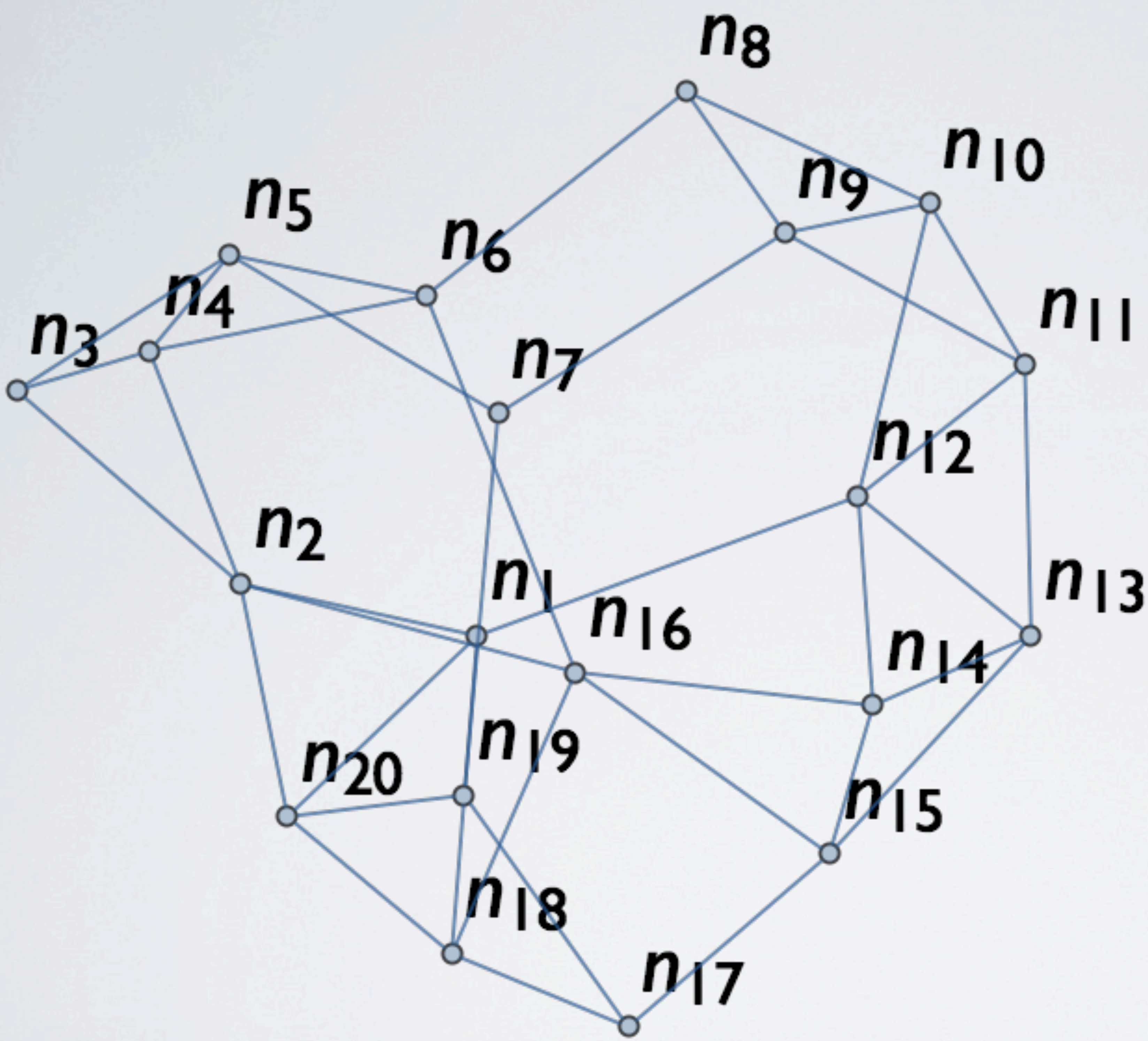
← ack

→ read

← 12

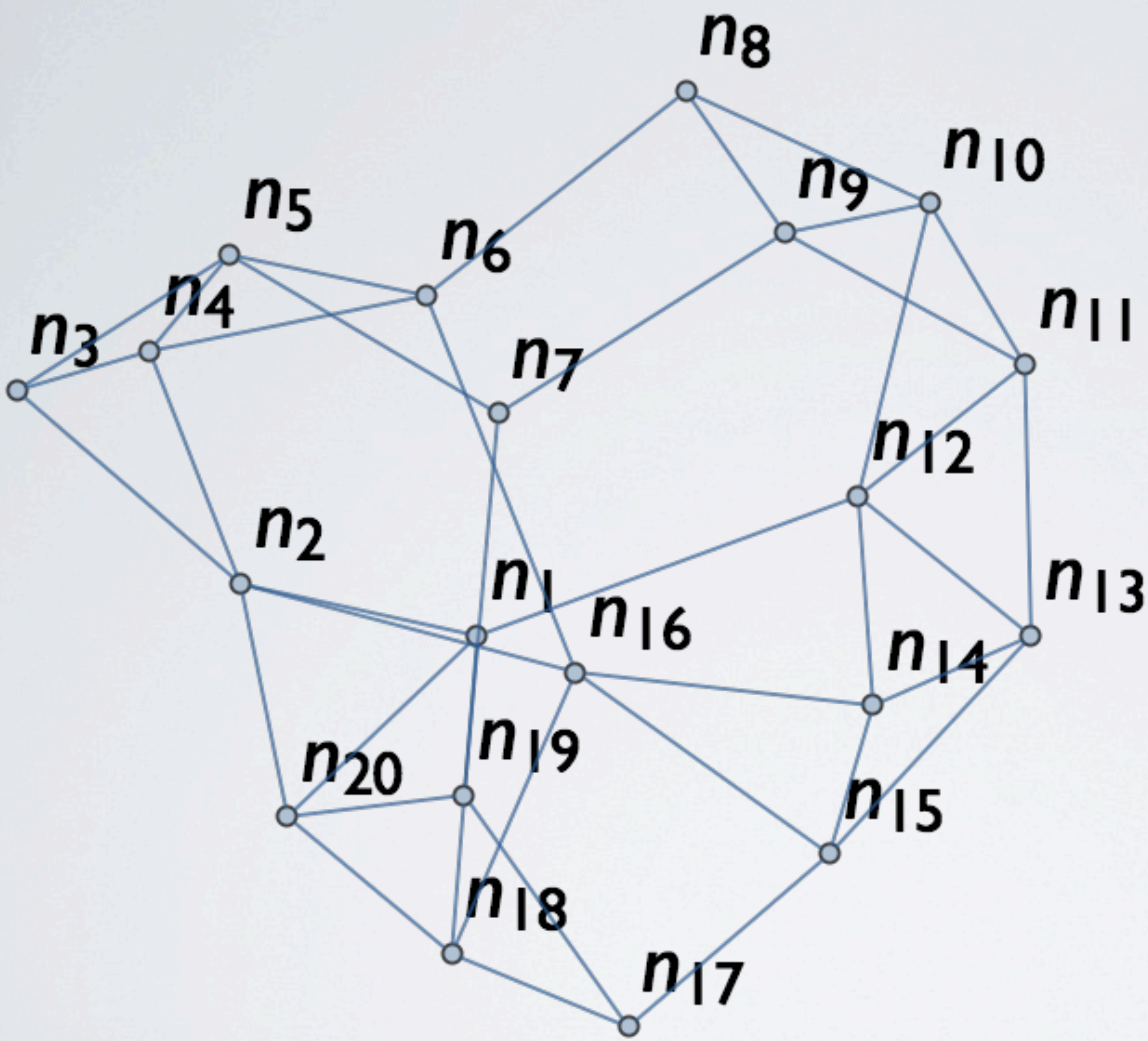
→ read

→ write 20

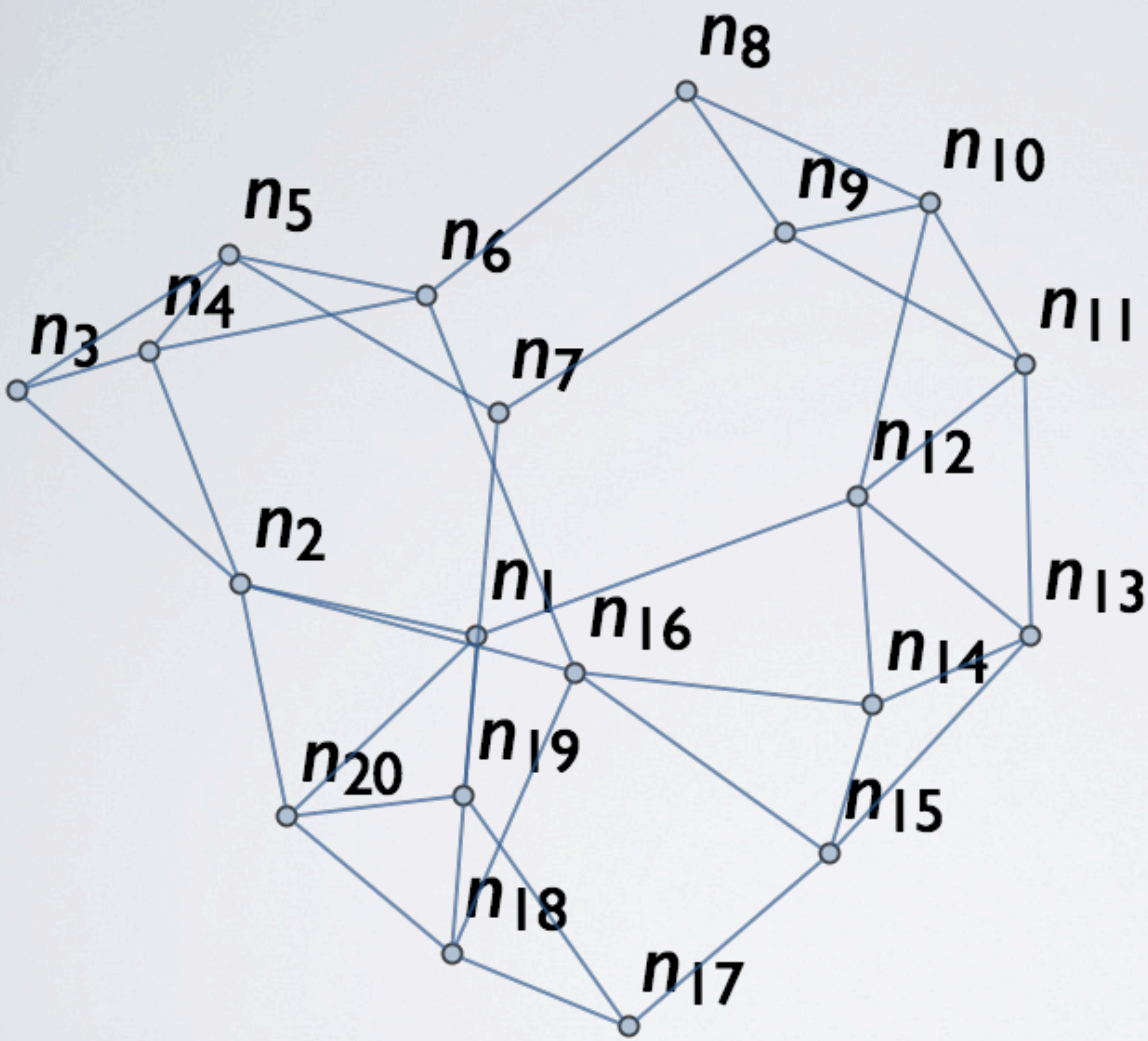


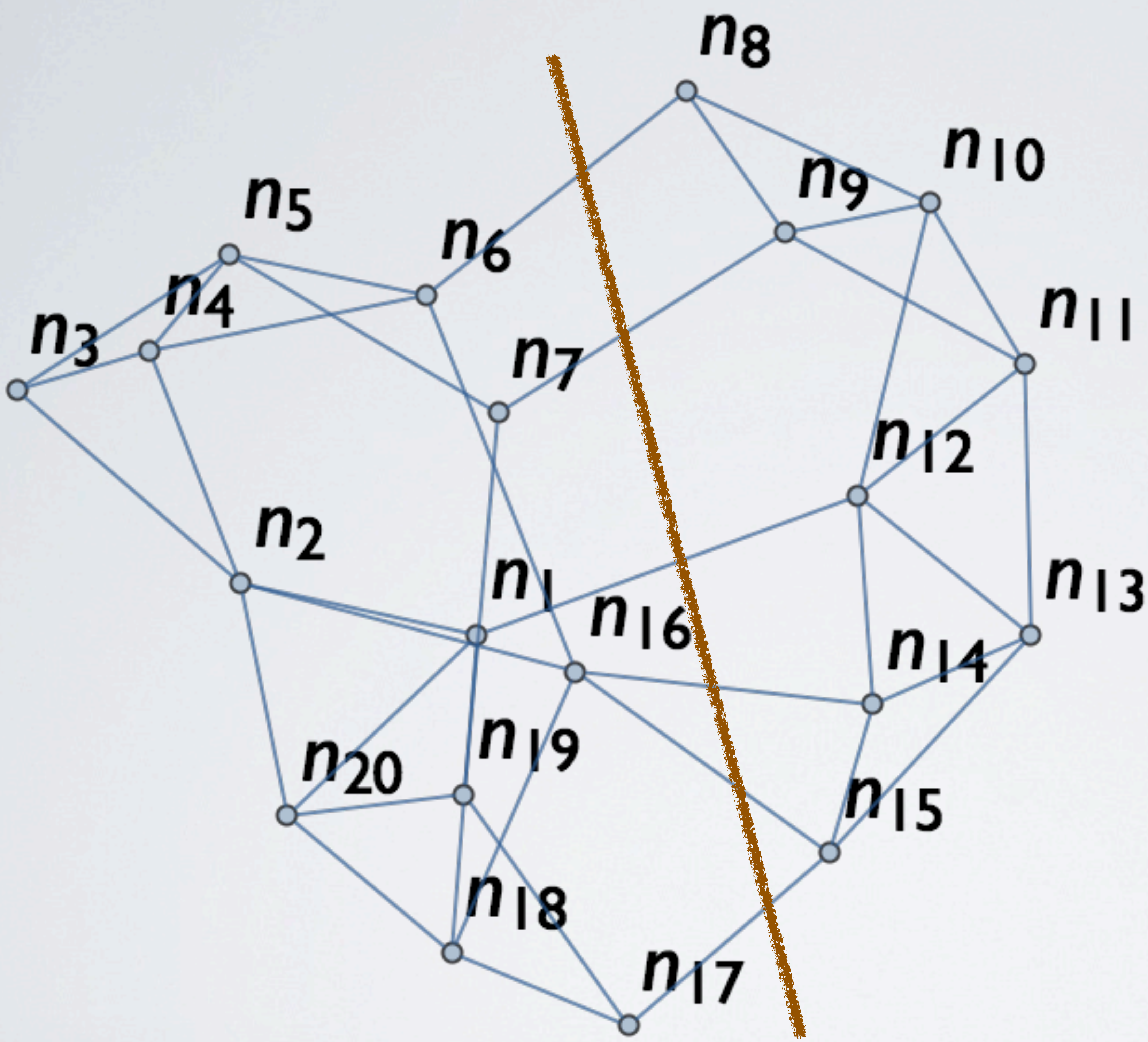
n_2

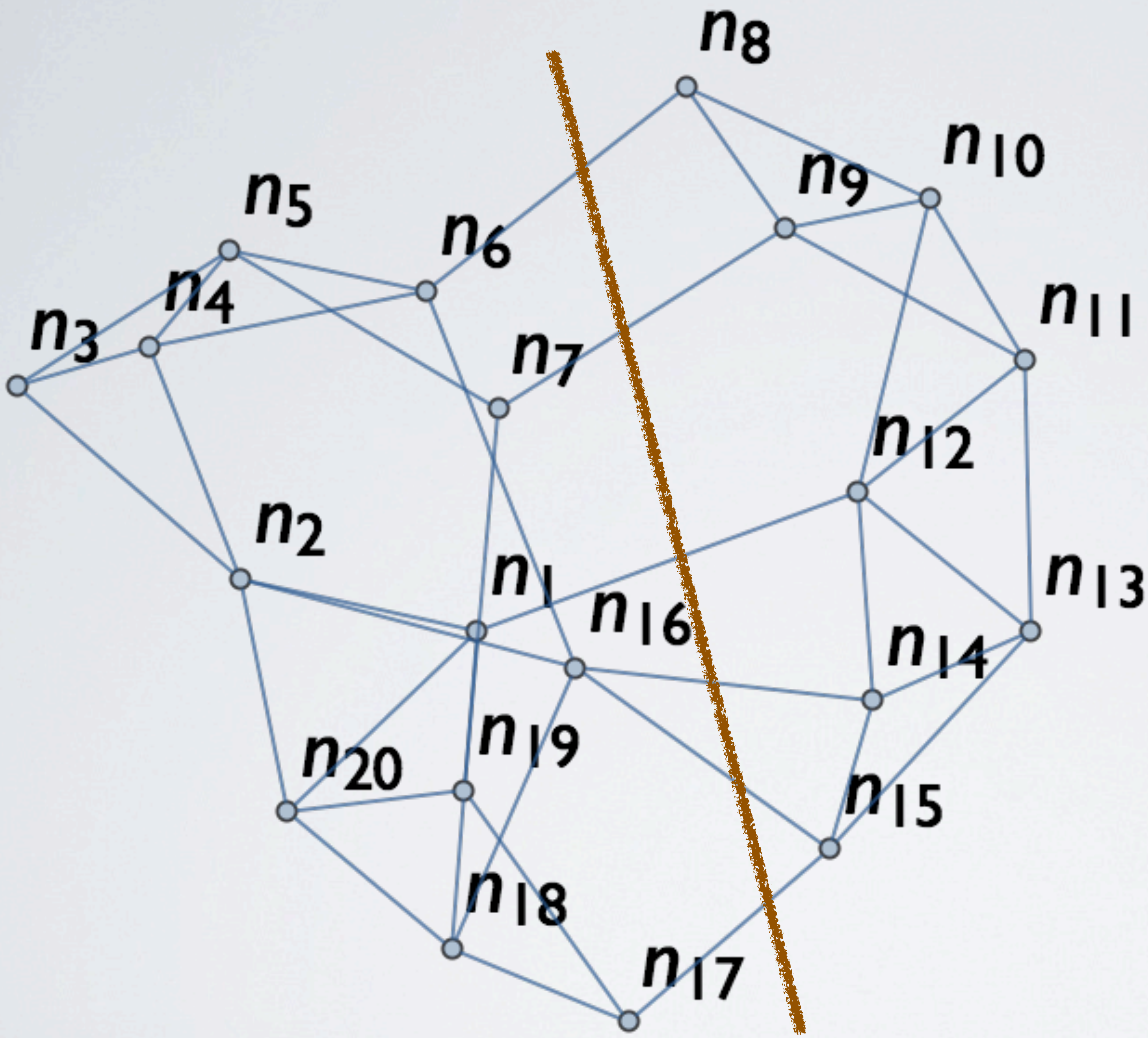
→	write	12
←	ack	
→	read	
←	12	
→	read	
→	write	20


n_8
n_{10}
n_9
n_5
n_6
n_4
n_{11}
n_3
n_7
n_{12}
n_2
n_1
n_{13}
n_{16}
n_{14}
n_{19}
n_{20}
n_{15}
n_{18}
n_{17}

Partitioning

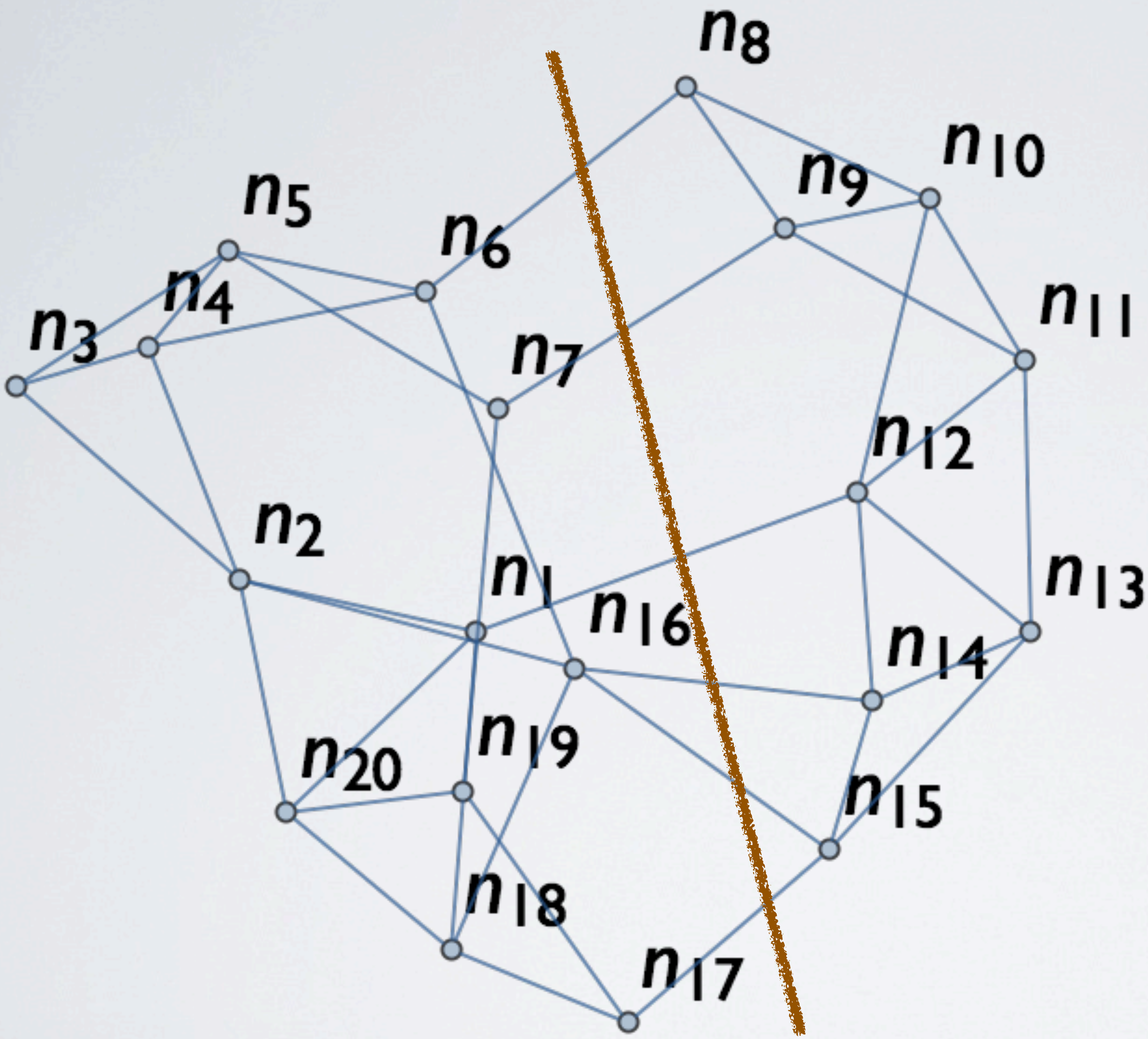

n_8
n_{10}
n_9
n_5
n_6
n_4
n_3
n_7
n_{11}
n_{12}
n_2
n_1
n_{13}
n_{16}
n_{14}
n_{19}
n_{20}
n_{15}
n_{18}
n_{17}


n_8
n_{10}
n_9
n_5
n_6
n_4
n_{11}
n_3
n_7
n_{12}
n_2
n_1
n_{13}
n_{16}
n_{14}
n_{20}
n_{19}
n_{15}
n_{18}
n_{17}



$G_1 = \{n_1, n_2, n_3, n_4, n_5, n_6, n_7, n_{16}, n_{17}, n_{18}, n_{19}, n_{20}\}$

$G_2 = \{n_8, n_9, n_{10}, n_{11}, n_{12}, n_{13}, n_{14}, n_{15}\}$



$G_1 = \{n_1, n_2, n_3, n_4, n_5, n_6, n_7, n_{16}, n_{17}, n_{18}, n_{19}, n_{20}\}$

$G_2 = \{n_8, n_9, n_{10}, n_{11}, n_{12}, n_{13}, n_{14}, n_{15}\}$

Theorem

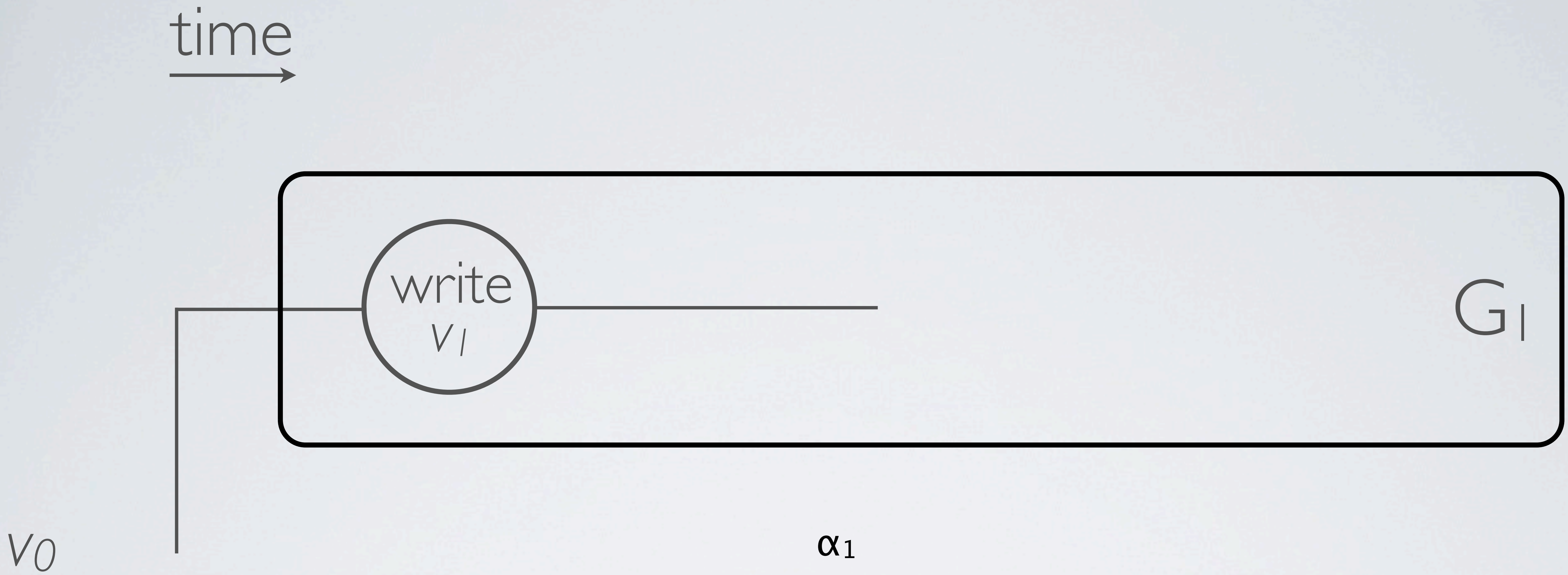
Shared atomic object

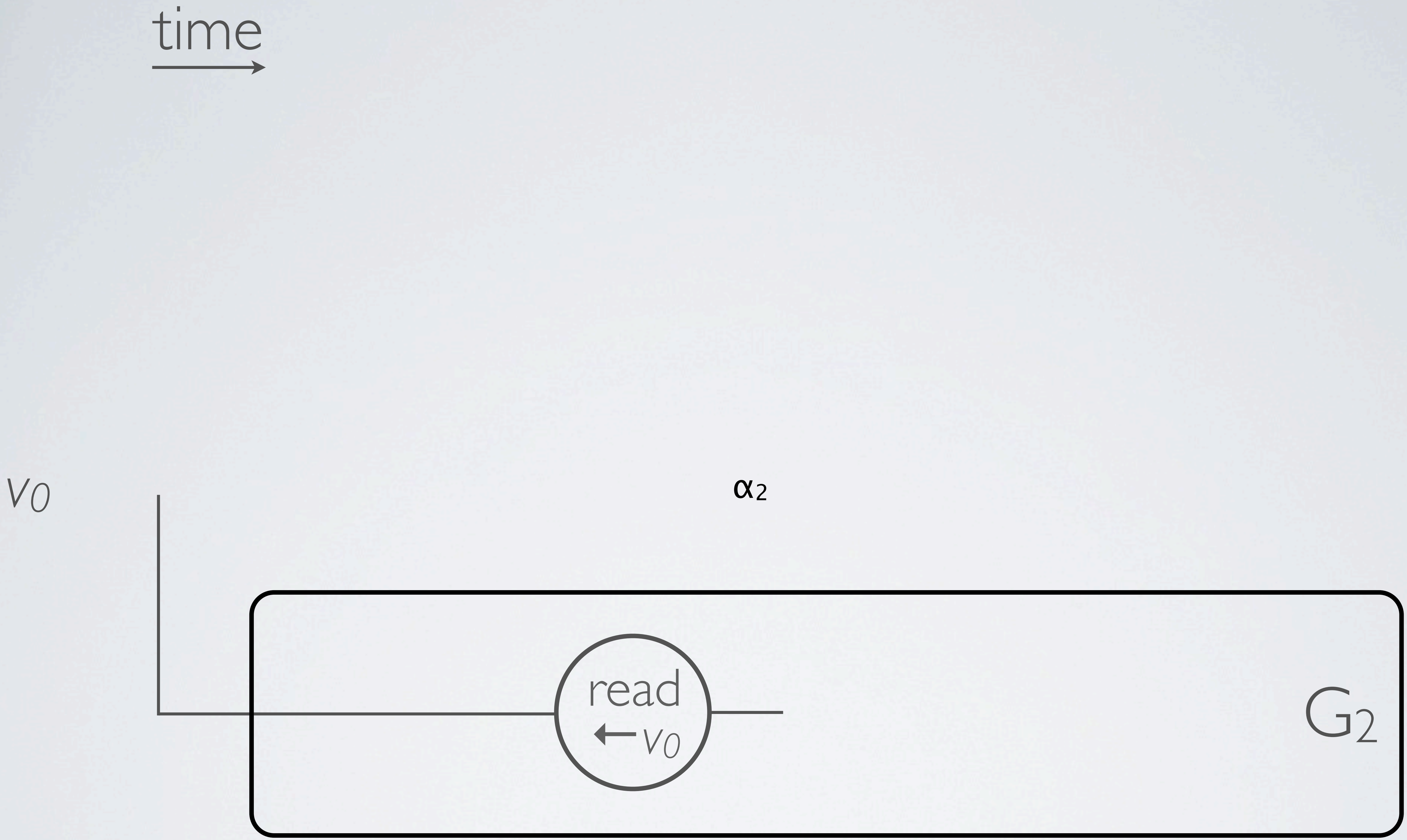
Asynchronous message-passing network

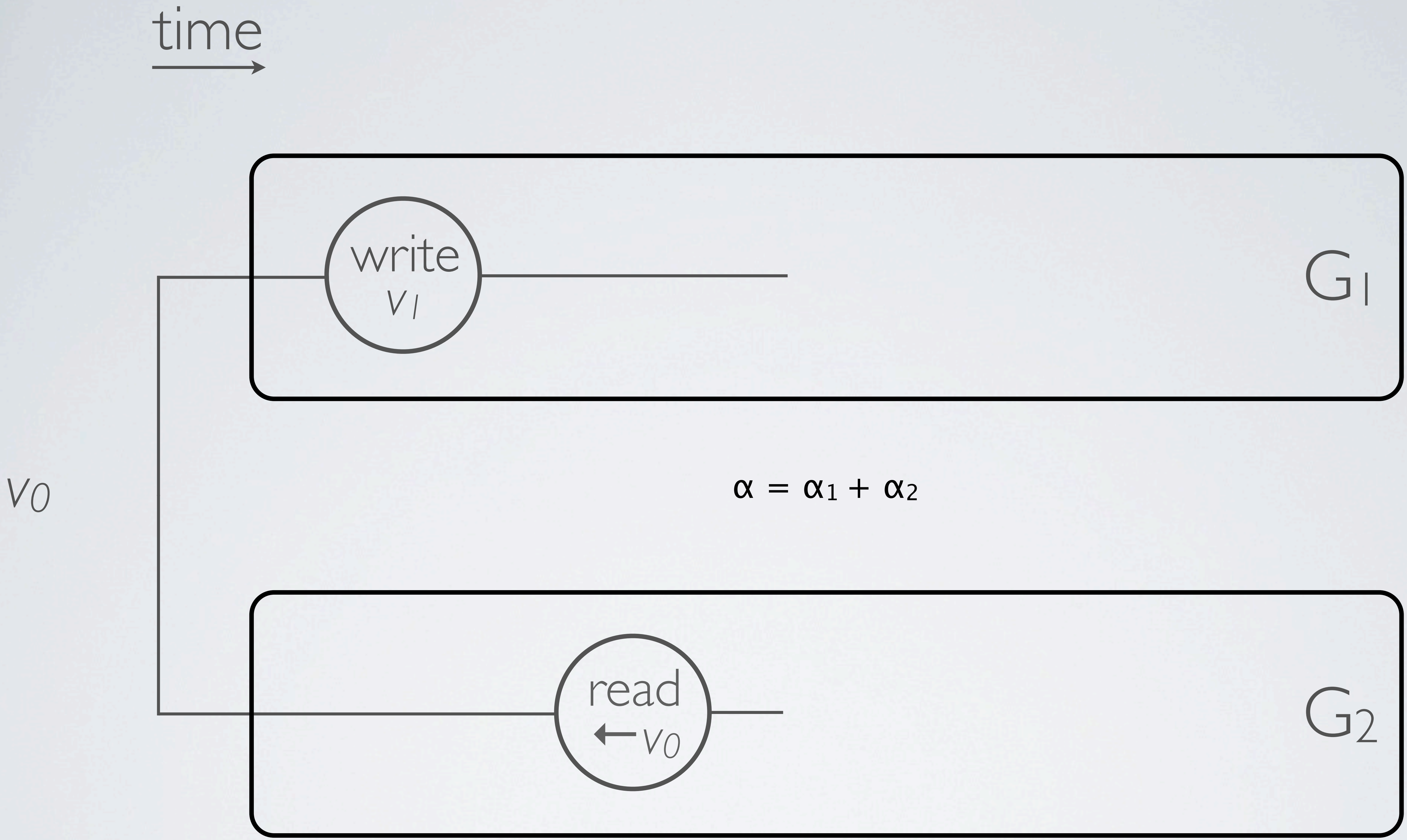
Network divided into $\{G_1, G_2\}$

All messages between G_1 and G_2 are lost

Suppose algorithm A meets all 3 of C, A, & P.


time
write
v1
G1
v0
α1


time
v_0
α_2
read
$\leftarrow v_0$
G_2


time
write
v_1
G_1
v_0
$\alpha = \alpha_1 + \alpha_2$
read
$\leftarrow v_0$
G_2

loophole

noun

loophole
noun

1. A way of *escaping a difficulty*, especially an omission or ambiguity in the wording of a contract or law that provides a means of *evading compliance*.

loophole

noun

1. A way of *escaping a difficulty*, especially an omission or ambiguity in the wording of a contract or law that provides a means of *evading compliance*.

2. A small hole or slit in a wall, especially one through which small arms may be fired.

Loophole I

HQ9+

H Prints "Hello, World!"

H	Prints "Hello, World!"
Q	Prints source text

H	Prints "Hello, World!"
Q	Prints source text
9	Prints lyrics to 99

H	Prints "Hello, World!"
Q	Prints source text
9	Prints lyrics to 99
+	Increments the register

Distributed HQ9+

H	Prints "Hello, World!"
Q	Prints source text
9	Prints lyrics to 99 bottles

H Prints "Hello, World!"

Q Prints source text

9 Prints lyrics to 99 bottles

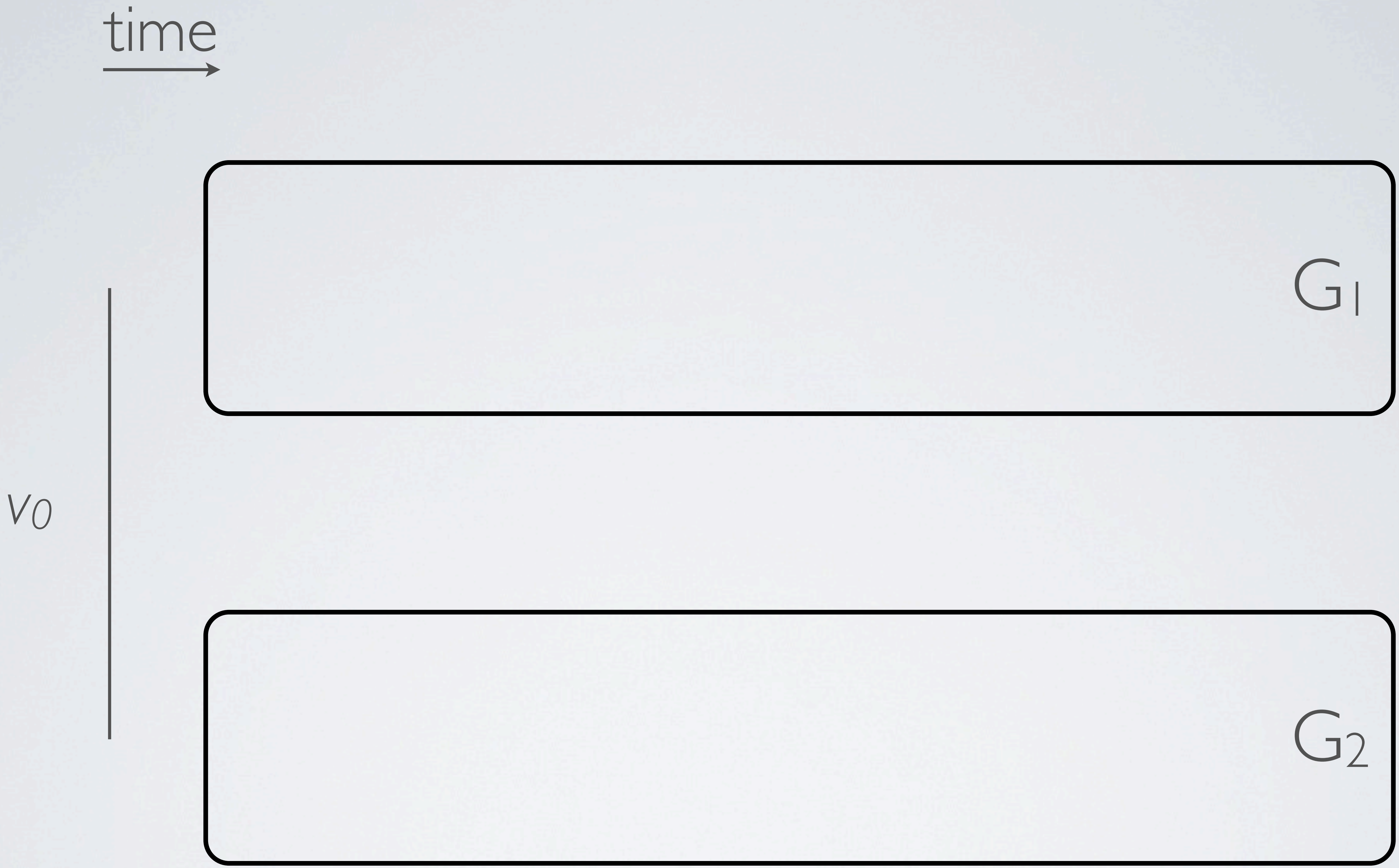
\+ Increments the ***distributed*** register

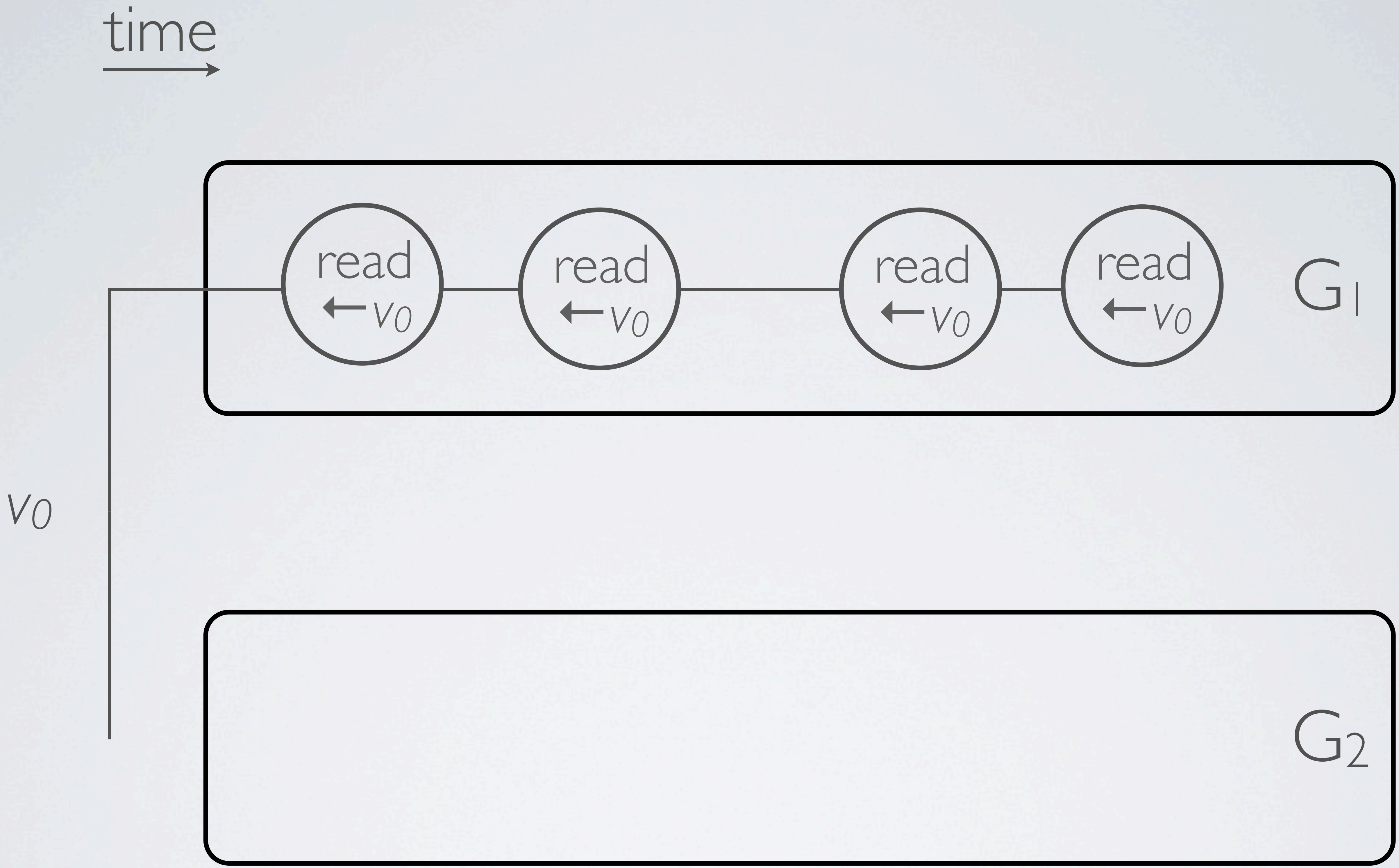
Loophole 2

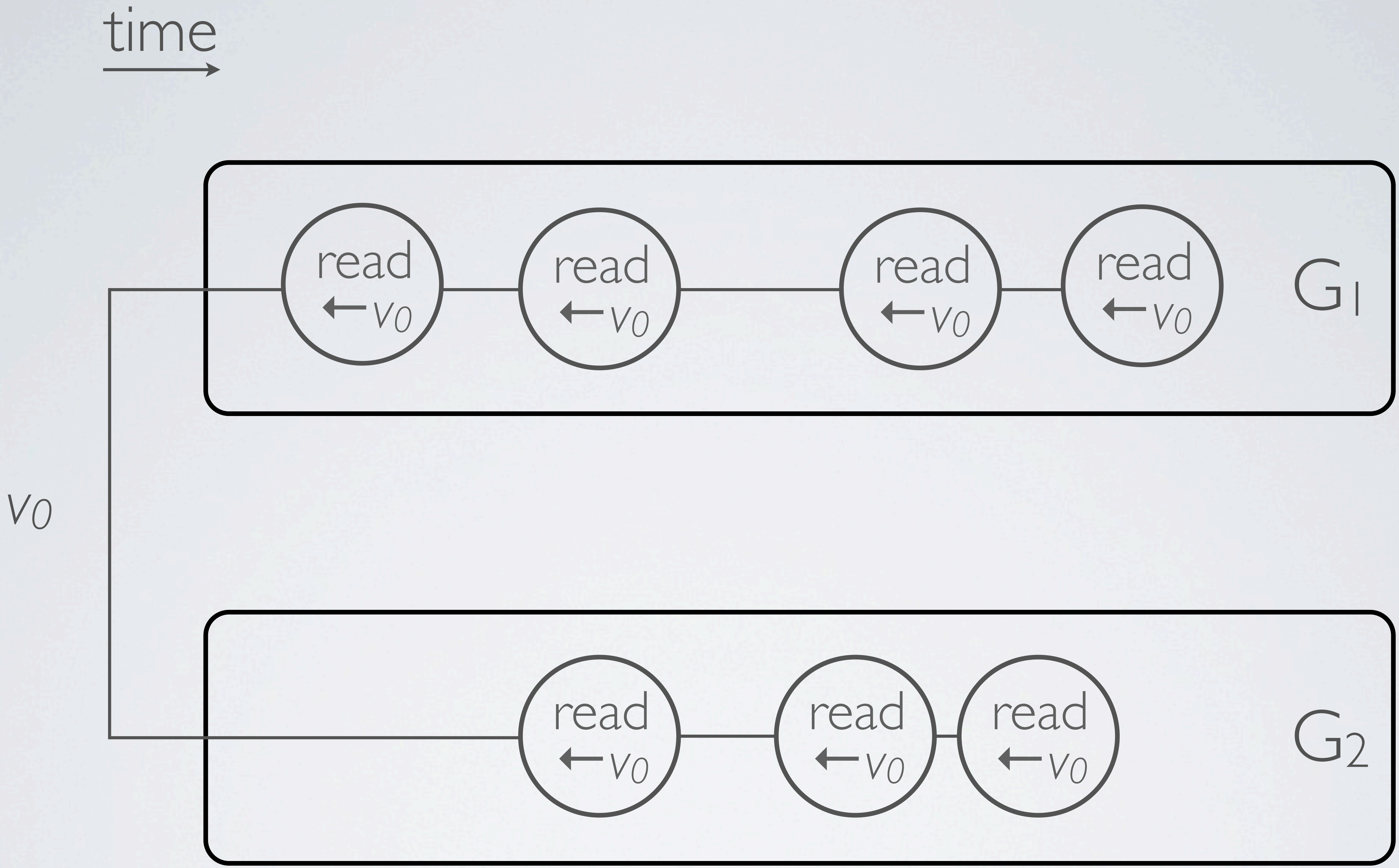
Write Once,
Immutable Thereafter

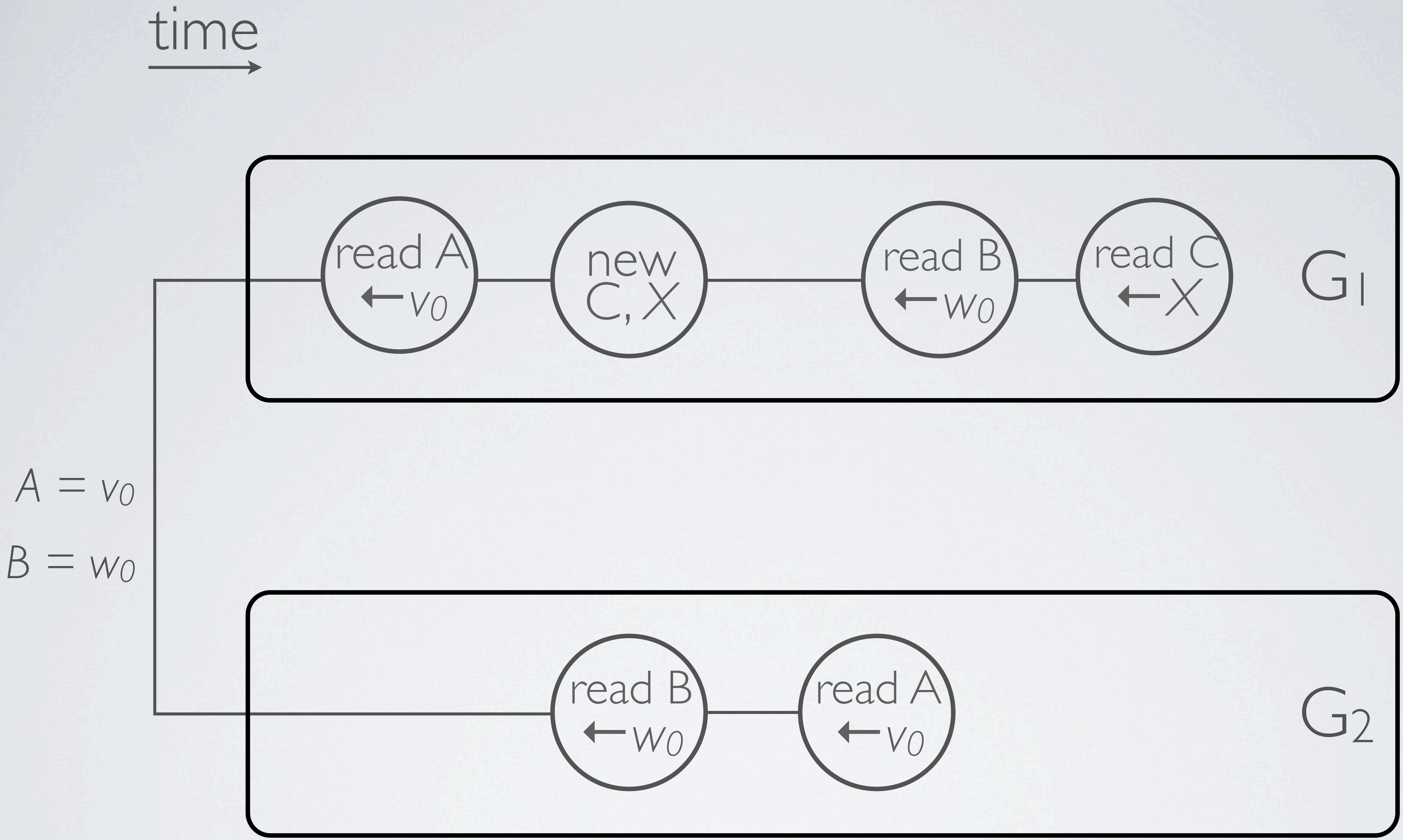
"Reading from immutable data is really fun, easy, and trivially consistent."

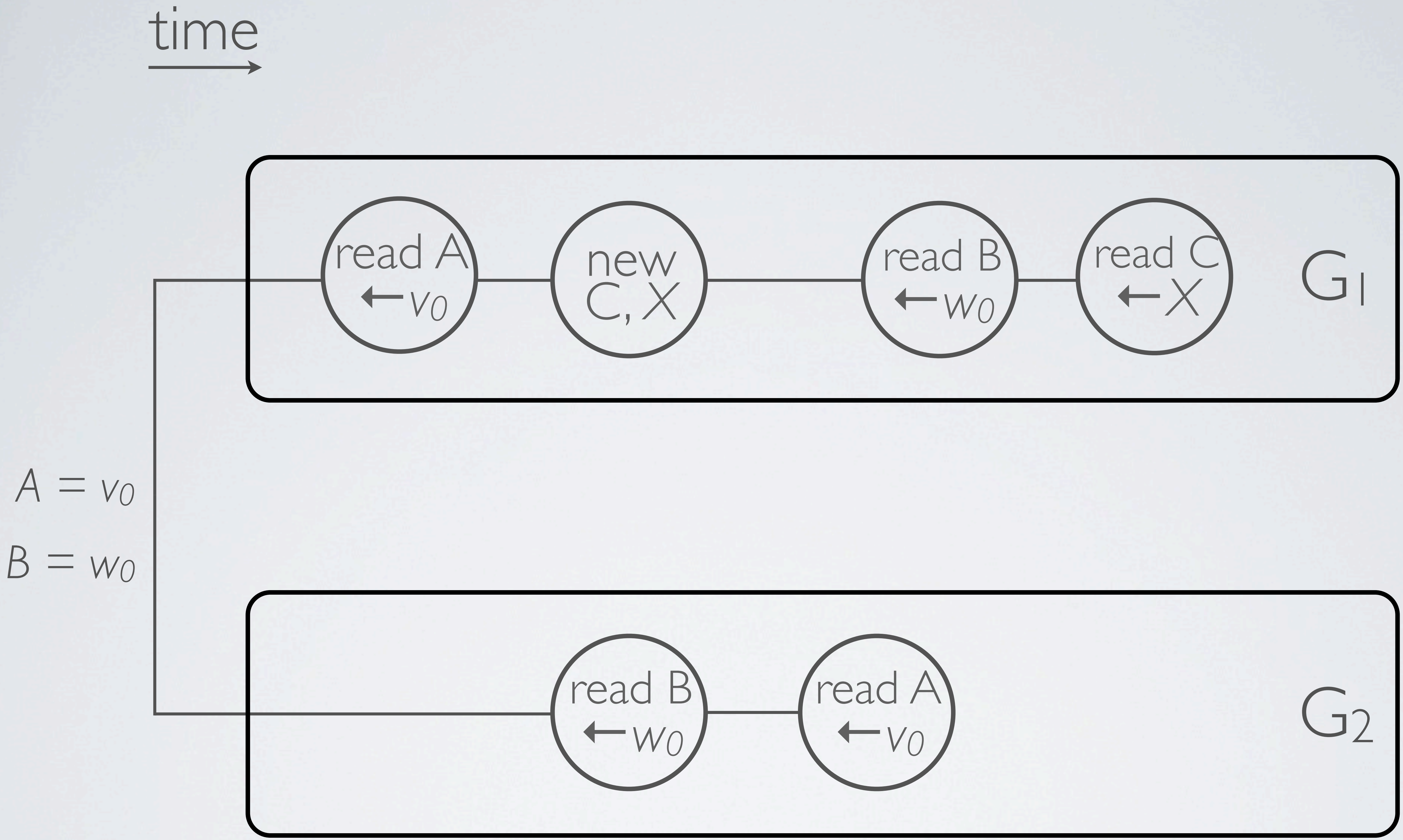
-- Eric Brewer, about an hour ago


time
G_1
v_0
G_2


time
read
$\leftarrow v_0$
read
$\leftarrow v_0$
read
$\leftarrow v_0$
read
$\leftarrow v_0$
G_1
v_0
G_2


time
v_0
read $\leftarrow v_0$
read $\leftarrow v_0$
read $\leftarrow v_0$
read $\leftarrow v_0$
G_1
read $\leftarrow v_0$
read $\leftarrow v_0$
read $\leftarrow v_0$
G_2


time
read A ← v_0
new C, X
read B ← w_0
read C ← X
G_1
$A = v_0$
$B = w_0$
read B ← w_0
read A ← v_0
G_2


time
read A ← v_0
new C, X
read B ← w_0
read C ← X
G_1
$A = v_0$
$B = w_0$
read B ← w_0
read A ← v_0
G_2

A bit of trickery?

Loophole 3

An older definition of consistency

The data base consists of entities which are related in certain ways. These relationships are best thought of as *assertions* about the data.

Examples of such assertions are:

"Names is an index for Telephone_numbers."

"The value of Count_of_X gives the number of employees in department X."

The data base is said to be *consistent* if it satisfies all its assertions. In some cases, the data base must become temporarily inconsistent in order to transform it to a new consistent state.

From "Granularity of Locks and Degrees of Consistency in a Shared Data Base",
J.N. Gray, R.A. Lorie, G.R. Putzolu, I.L. Traiger,

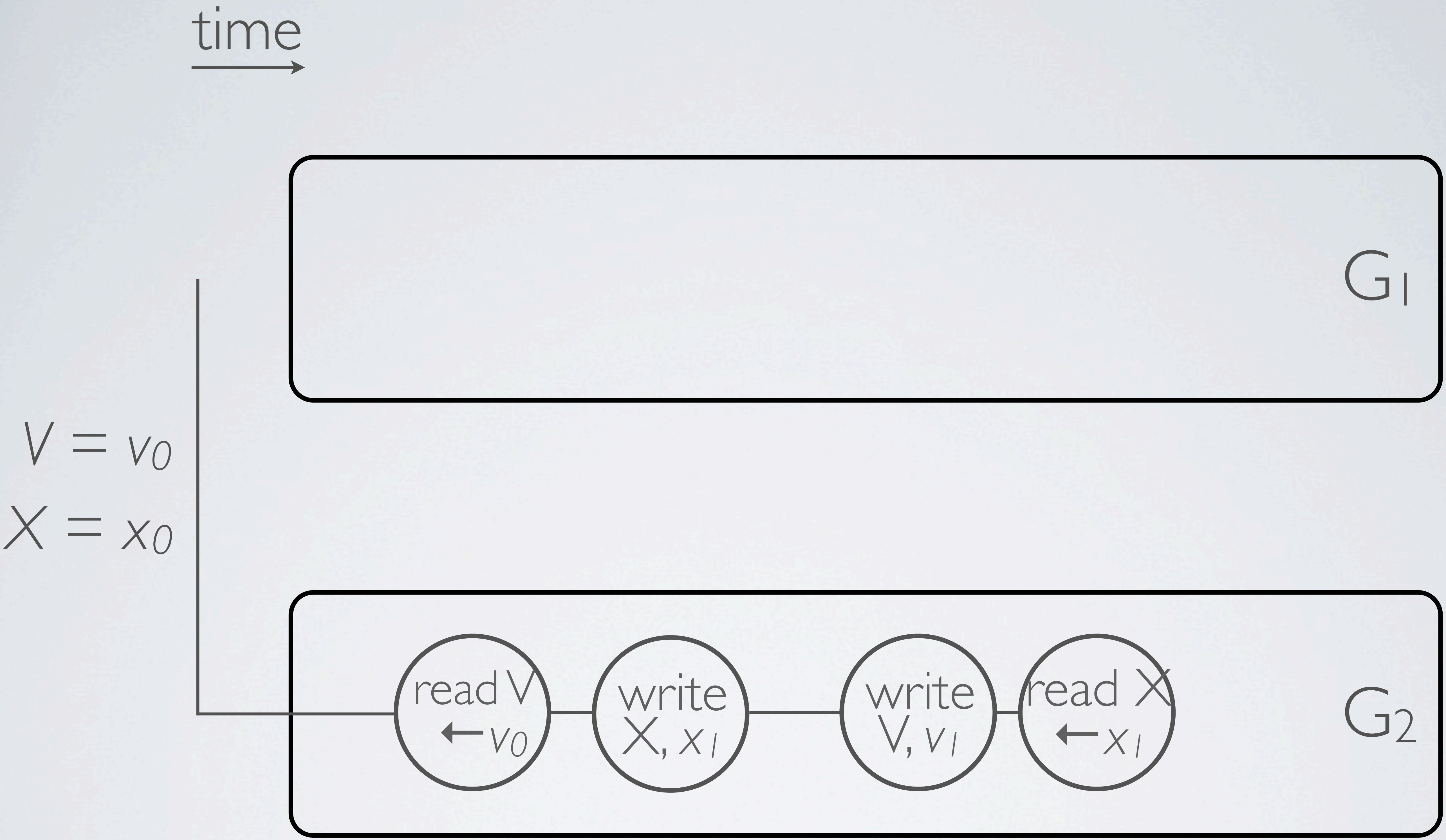
The data base is said to be *consistent* if it satisfies all its assertions. In some cases, the data base must become temporarily inconsistent in order to transform it to a new consistent state.

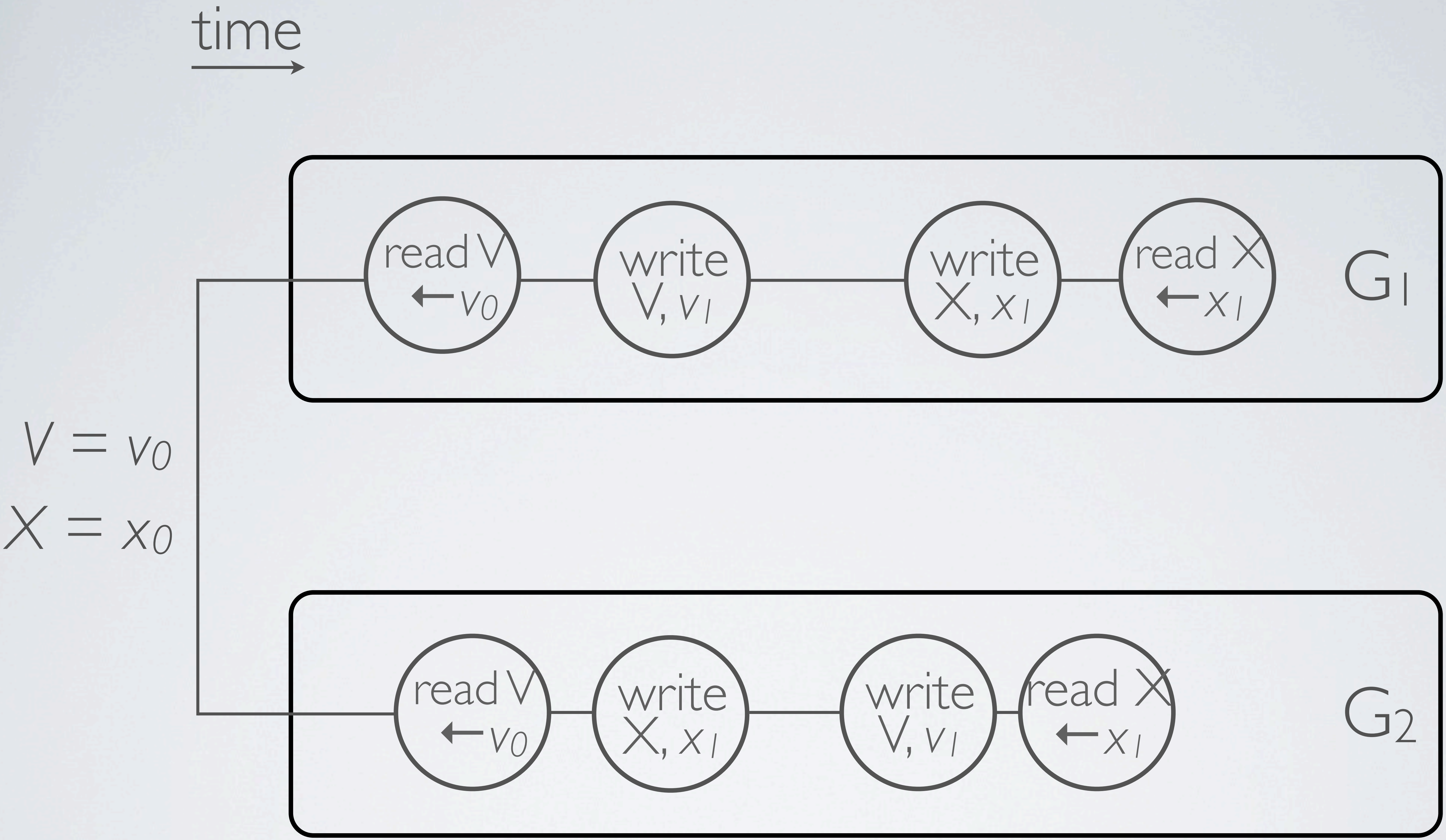
From "Granularity of Locks and Degrees of Consistency in a Shared Data Base",
J.N. Gray, R.A. Lorie, G.R. Putzolu, I.L. Traiger, **1976**

Consistency is a predicate C on entities and their values. The predicate is generally not known to the system but is embodied in the structure of the transactions.

From "Transactions and Consistency in Distributed Database Systems",
I.L. Traiger, J.N. Gray, C.A. Galtieri, and B.G. Lindsay, 1982

Can this kind of consistency be maintained in a distributed system?


time
G1
$V = v_0$
$X = x_0$
read V ← v_0
write X, x_1
write V, v_1
read X ← x_1
G2


time
read V $\leftarrow v_0$
write V, v_1
write X, x_1
read X $\leftarrow x_1$
G_1
$V = v_0$
$X = x_0$
read V $\leftarrow v_0$
write X, x_1
write V, v_1
read X $\leftarrow x_1$
G_2

C
R
D
T

Commutative
Replicated
Data
Type

Loophole 4

Partition A: $<C_a, G_a, a_1, a_2, \ldots, a_n>$

Partition A: $<C_a, G_a, a_1, a_2, \ldots, a_n>$

C_a Consistency predicate over $a_1 \ldots a_n$

Partition A: $\langle C_a, G_a, a_1, a_2, \ldots, a_n \rangle$

C_a Consistency predicate over $a_1 \ldots a_n$

G_a Subset of nodes in network

Partition A: $<C_a, G_a, a_1, a_2, \ldots, a_n>$

C_a Consistency predicate over $a_1 \ldots a_n$

G_a Subset of nodes in network

a_i Value of variable i

Partition A: $<C_a, G_a, a_1, a_2, \ldots, a_n>$

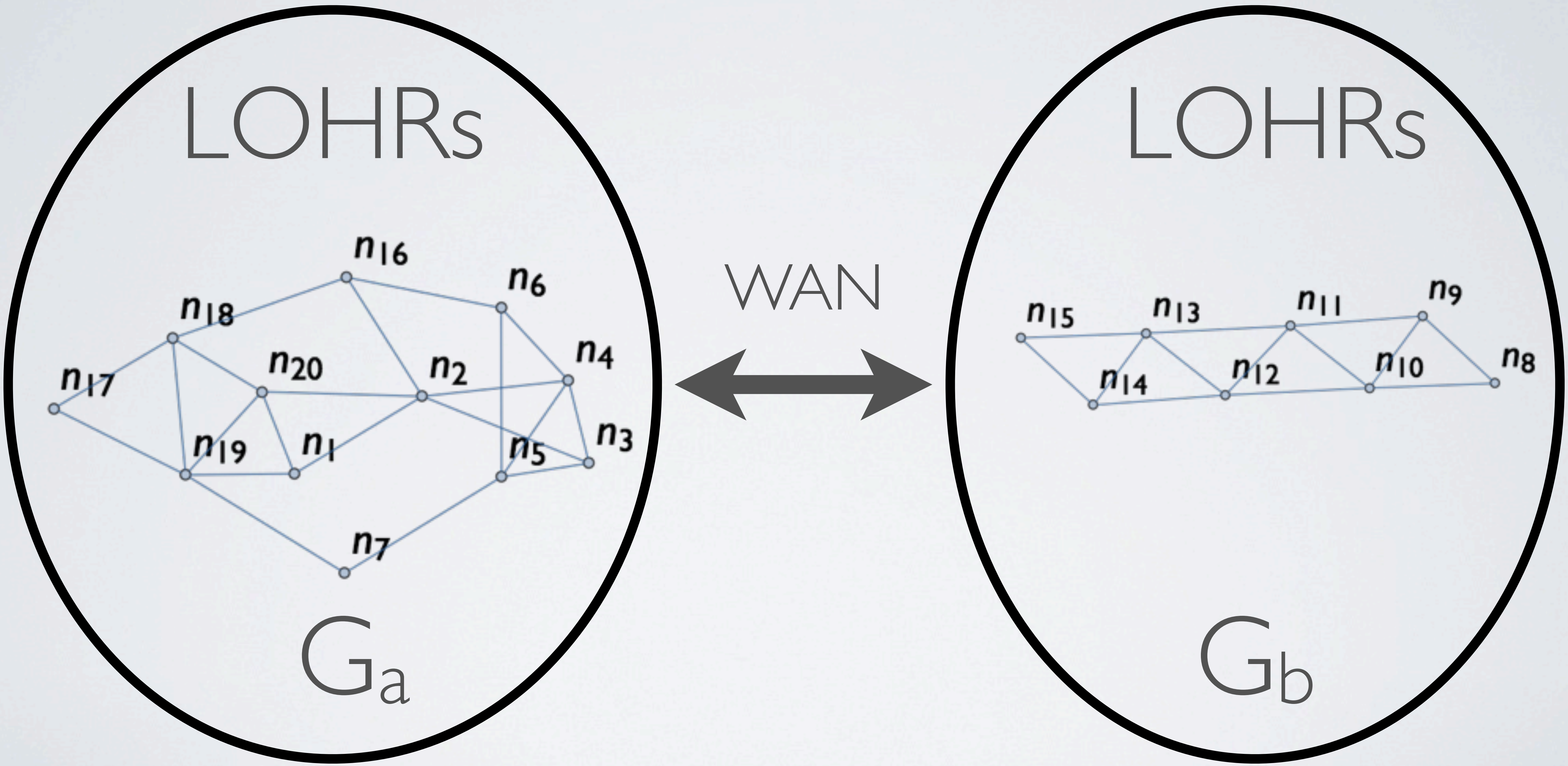
C_a	Consistency predicate over $a_1 \ldots a_n$
G_a	Subset of nodes in network
a_i	Value of variable i

Partition A: $\langle C_a, G_a, a_1, a_2, \ldots, a_n \rangle$

Partition A: $\langle C_a, G_a, a_1, a_2, \ldots, a_n \rangle$

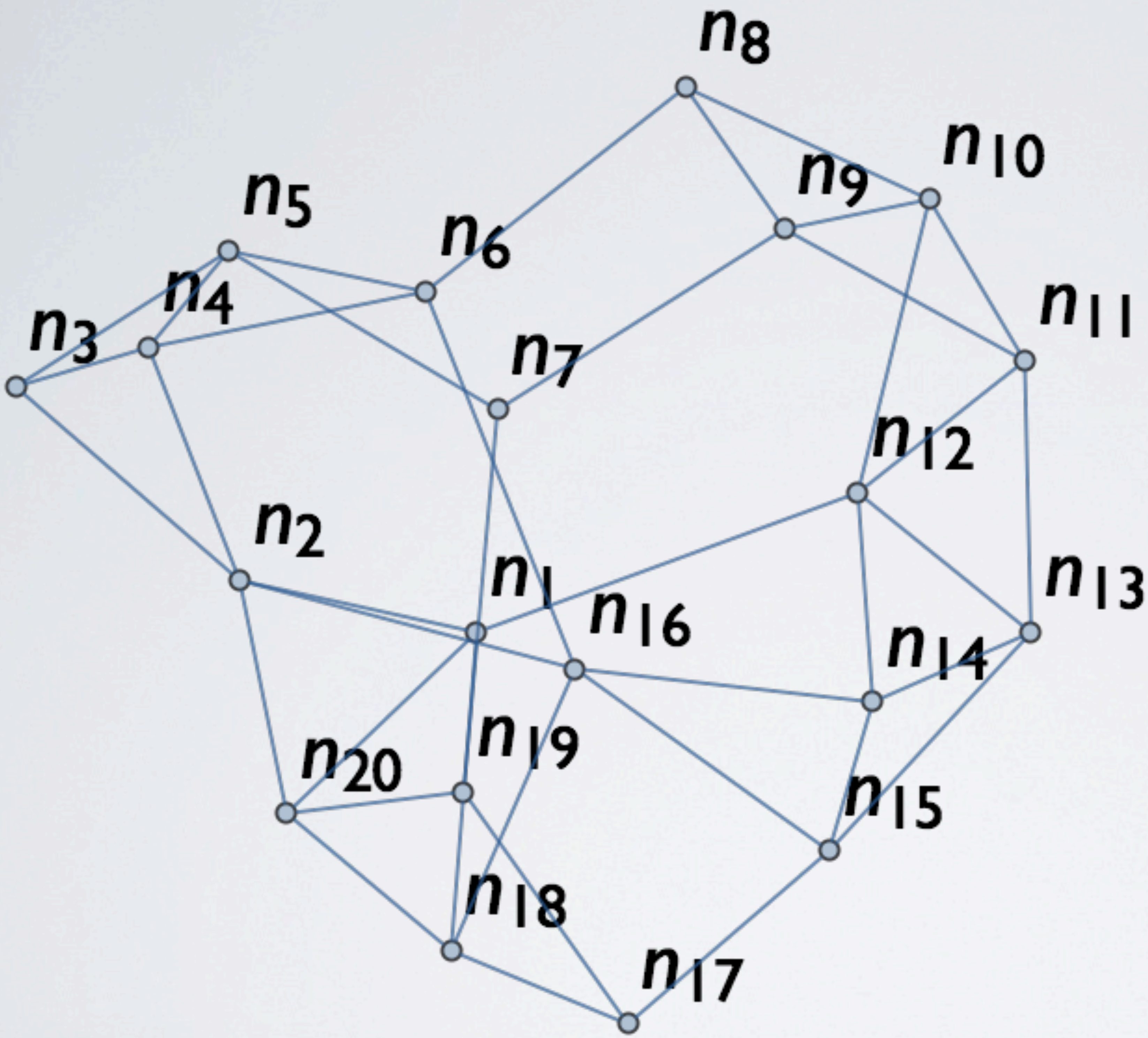
Partition A: $\langle C_a, G_a, a_1, a_2, \ldots, a_n \rangle$

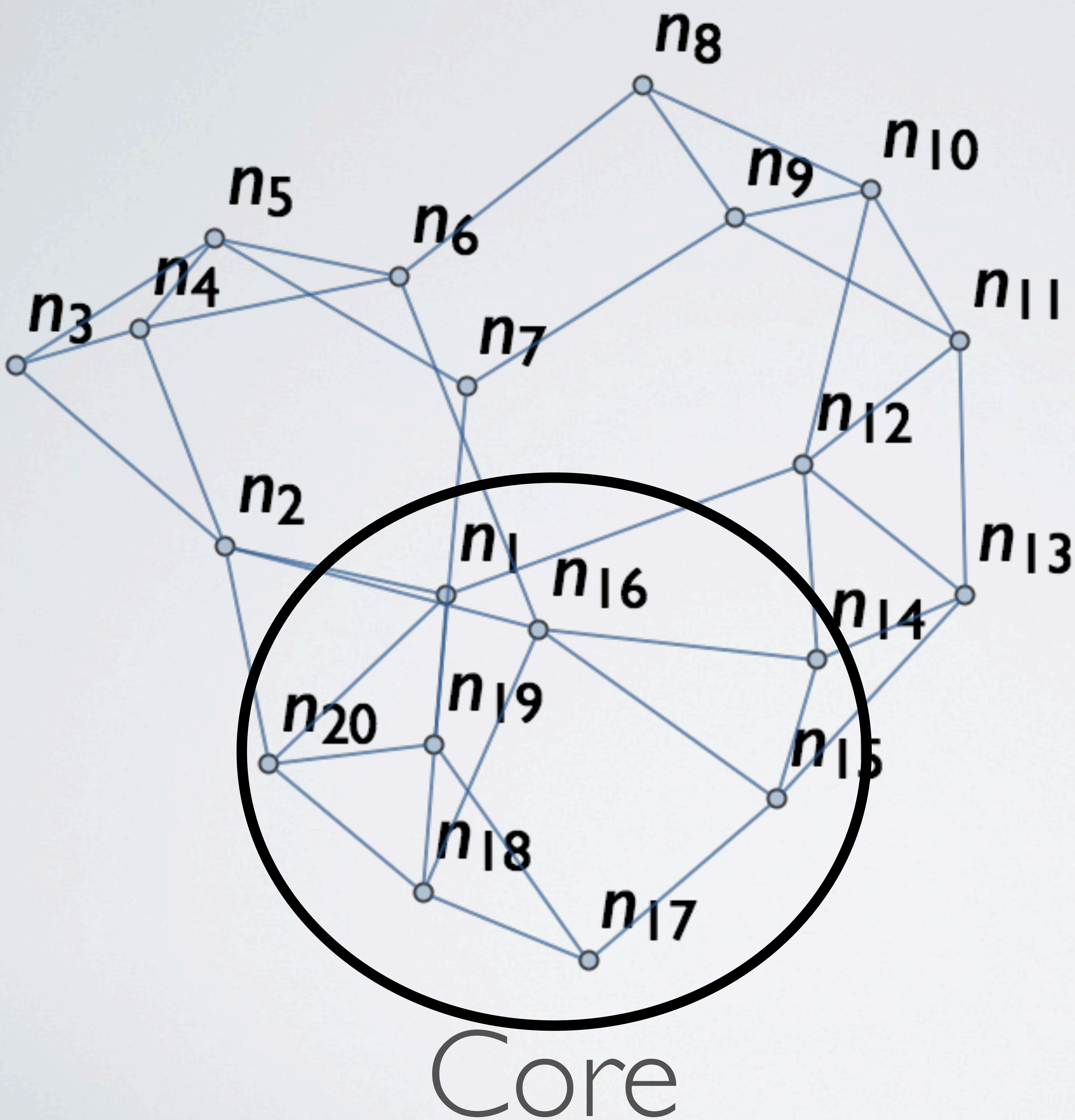
Partition B: $\langle C_b, G_b, b_1, b_2, \ldots, b_m \rangle$


LOHRs
n16
n6
n18
n4
n20
n2
n17
n19
n1
n5
n3
n7
Ga
WAN
LOHRs
n15
n13
n11
n9
n14
n12
n10
n8
Gb

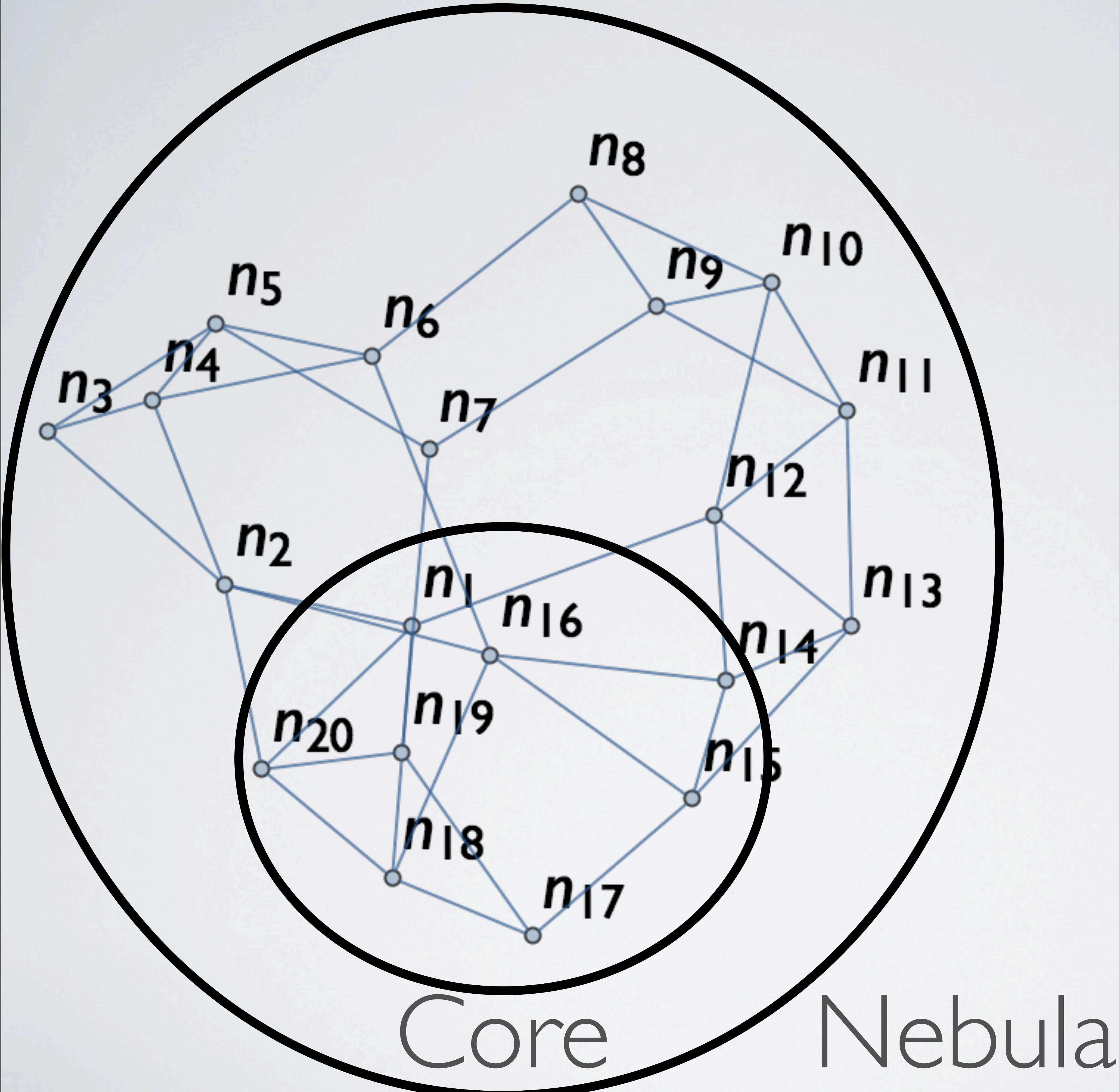
Loophole 5

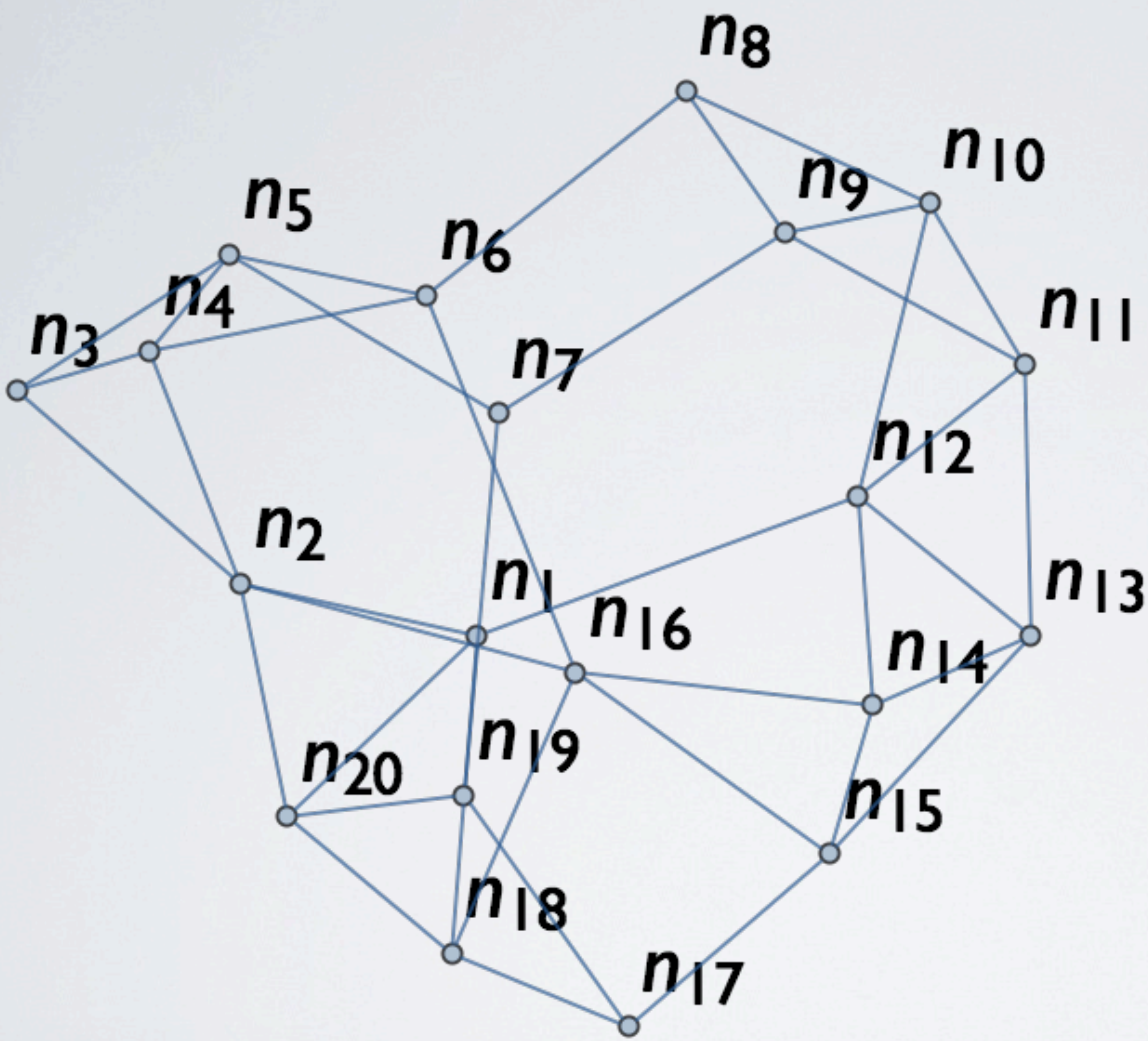
Bounded Consistency

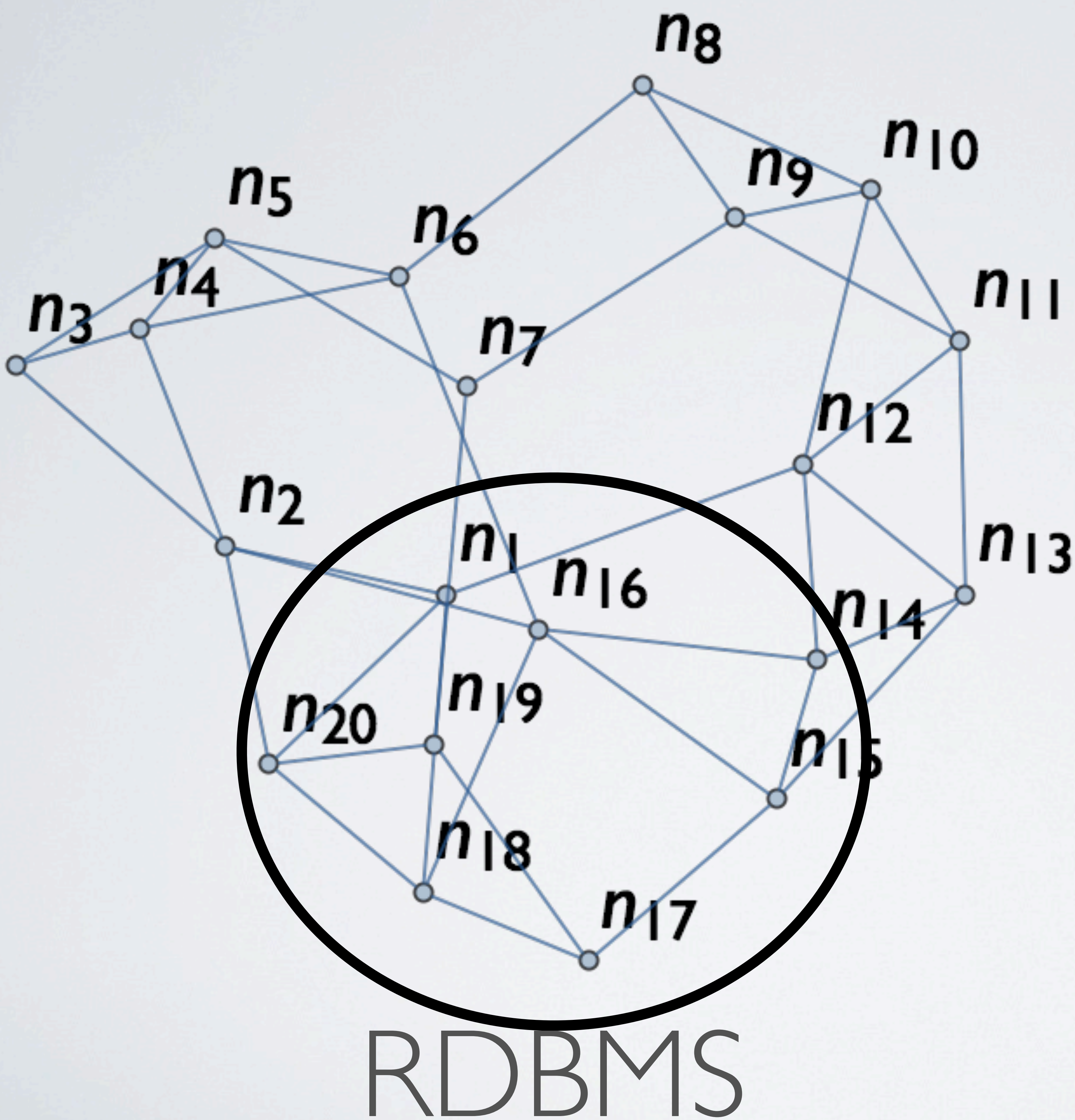

n_8
n_{10}
n_9
n_5
n_6
n_4
n_{11}
n_3
n_7
n_{12}
n_2
n_{13}
n_1
n_{16}
n_{14}
n_{19}
n_{20}
n_{15}
n_{18}
n_{17}

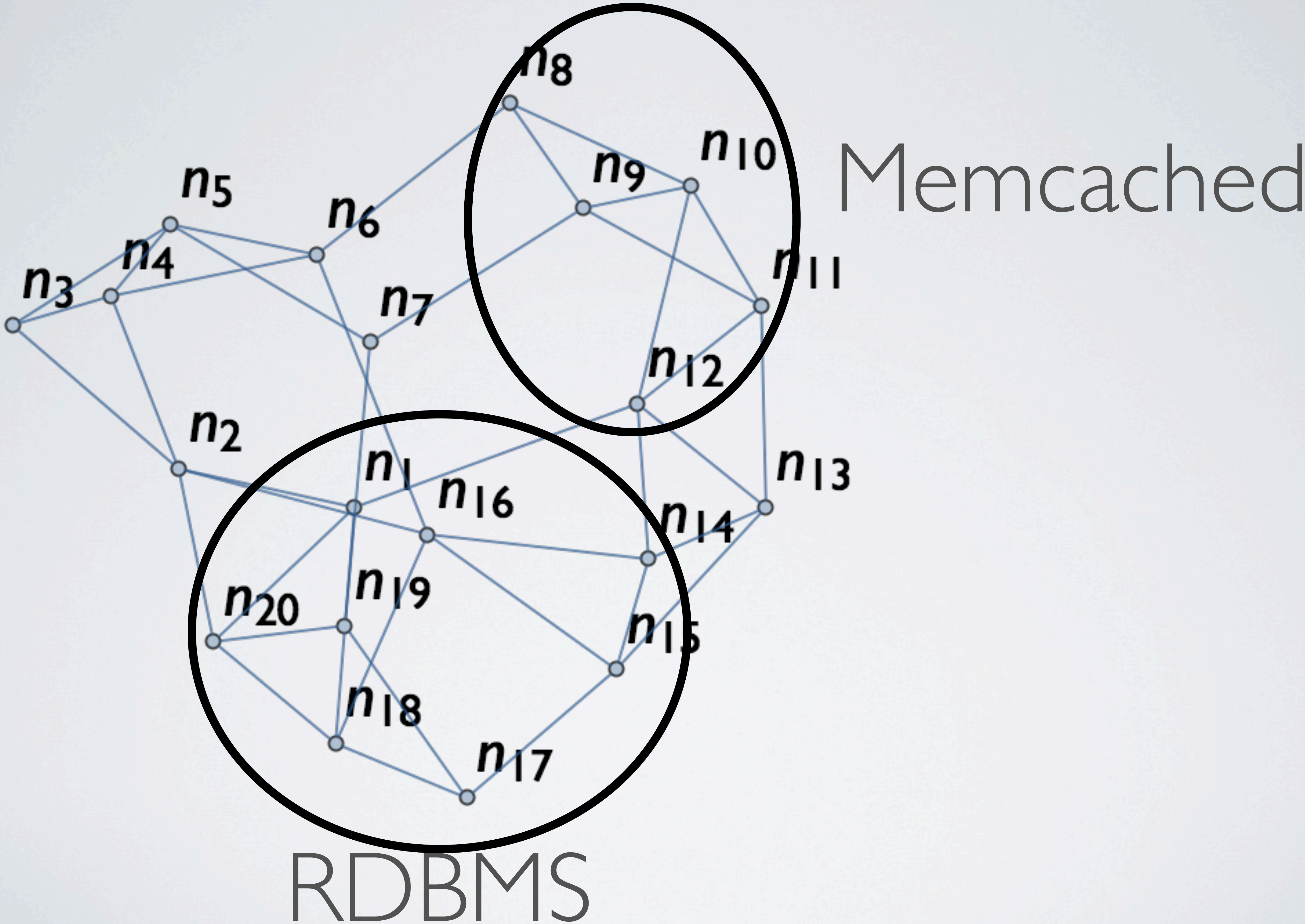


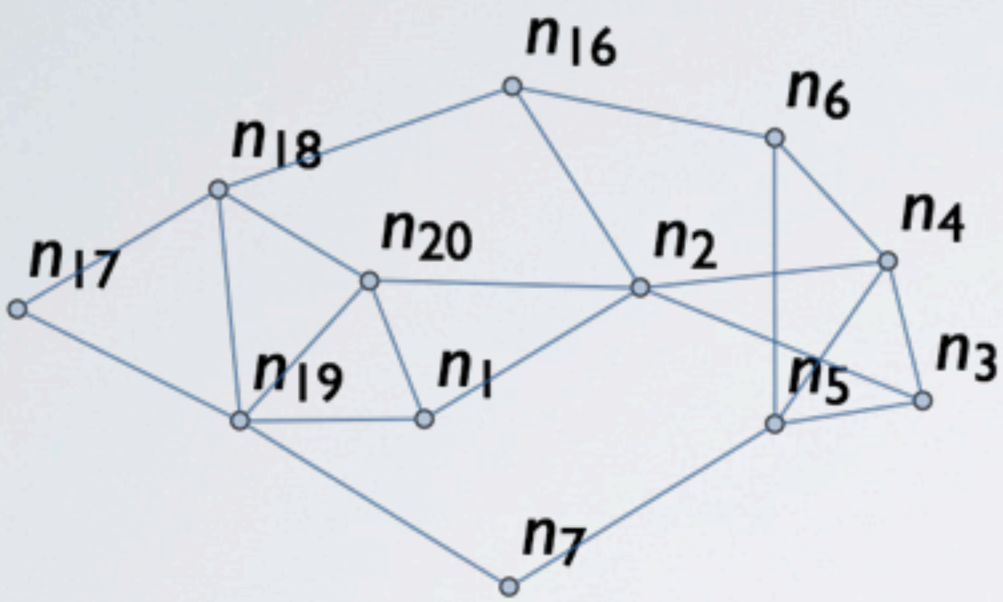
n_8
n_{10}
n_9
n_5
n_6
n_4
n_{11}
n_3
n_7
n_{12}
n_2
n_{13}
n_1
n_{16}
n_{14}
n_{19}
n_{20}
n_{15}
n_{18}
n_{17}
Core


n_8
n_{10}
n_9
n_5
n_6
n_4
n_{11}
n_3
n_7
n_{12}
n_2
n_{13}
n_1
n_{16}
n_{14}
n_{19}
n_{20}
n_{15}
n_{18}
n_{17}
Core
Nebula

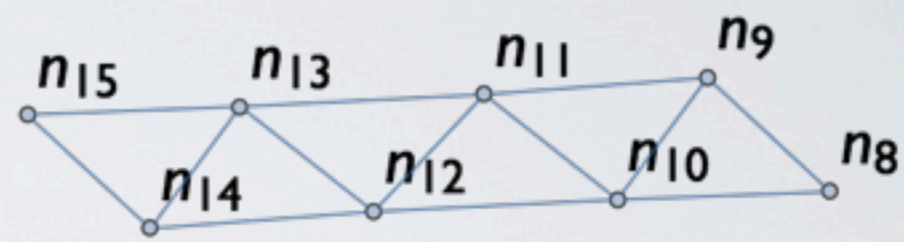

n_8
n_{10}
n_9
n_5
n_6
n_4
n_{11}
n_3
n_7
n_{12}
n_2
n_{13}
n_1
n_{16}
n_{14}
n_{19}
n_{20}
n_{15}
n_{18}
n_{17}


n8
n10
n9
n5
n6
n11
n4
n3
n7
n12
n2
n13
n1
n16
n14
n19
n20
n15
n18
n17
RDBMS

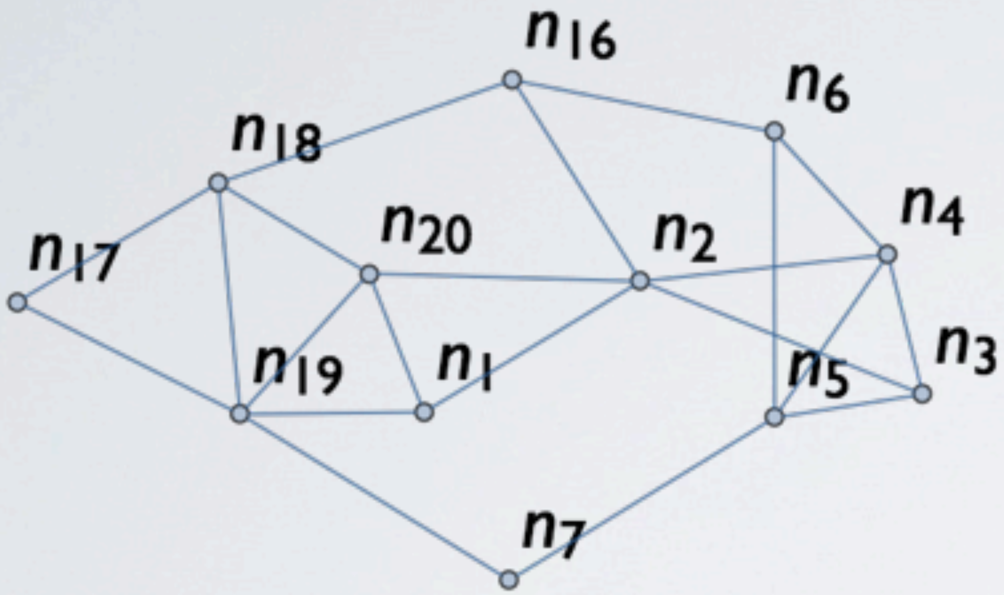

n_8
n_{10}
n_9
Memcached
n_5
n_6
n_4
n_{11}
n_3
n_7
n_{12}
n_2
n_{13}
n_1
n_{16}
n_{14}
n_{19}
n_{20}
n_{15}
n_{18}
n_{17}
RDBMS

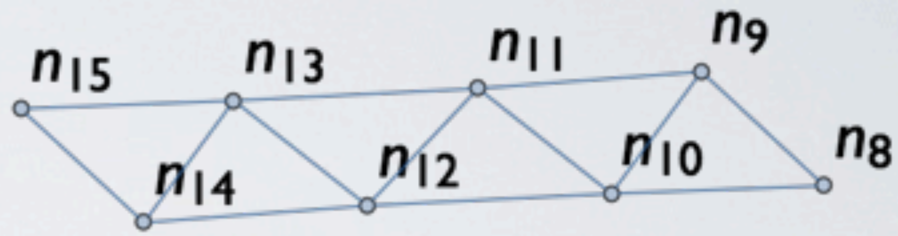


n_{16}
n_6
n_{18}
n_4
n_{20}
n_2
n_{17}
n_{19}
n_1
n_5
n_3
n_7



n_{15}
n_{13}
n_{11}
n_9
n_{14}
n_{12}
n_{10}
n_8





Item Display
A & P
Heavy caching

Bid History
C & P
Strong consistency

Loophole 6

Stop building distributed systems

Loophole 7

Get a better network!

Asynchronous message passing

Asynchronous message passing

That's UDP!

Semi-synchronous network

Lost messages are detected after time t (by a missed acknowledgement)

“Delayed-*t* Consistency”

A partial ordering P orders all writes, and all reads with respect to writes.

The value of every read is the one written by the previous write, where “previous” is under P.

The order in P is consistent with the order of read and write requests at each node.

If all messages are delivered and an operation θ

The value of every read is the one written by the previous write, where "previous" is under P.

The order in P is consistent with the order of read and write requests at each node.

If all messages are delivered and an operation θ completes before Φ begins, then Φ does not precede θ in P.

Assume an interval greater than t in which no messages are lost. Further assume that θ begins before the interval and Φ begins after the interval ends. Then Φ does not precede θ in P.

“‘Delayed-*t* Consistency”

"Eventual Consistency"

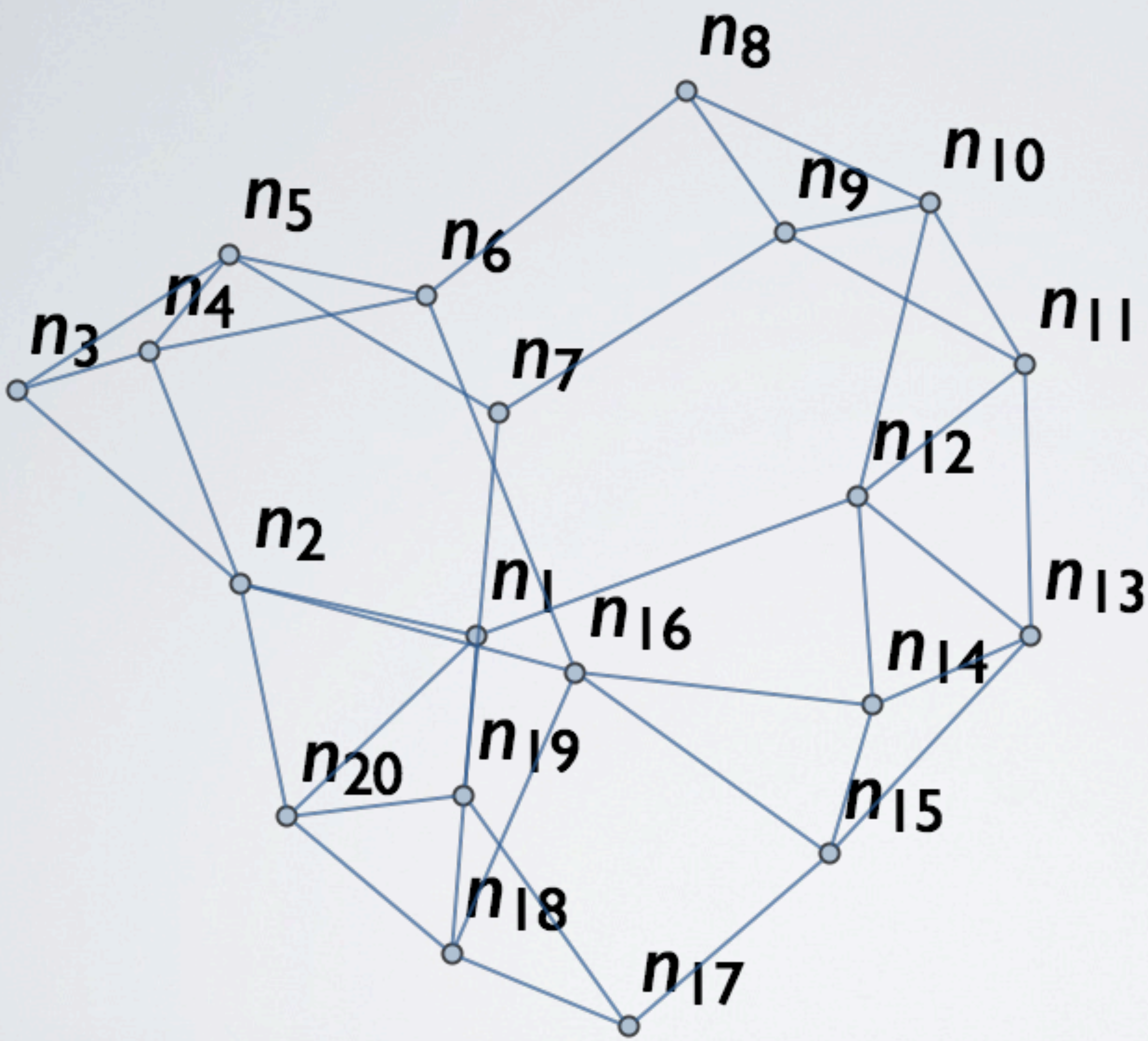
Loophole 7

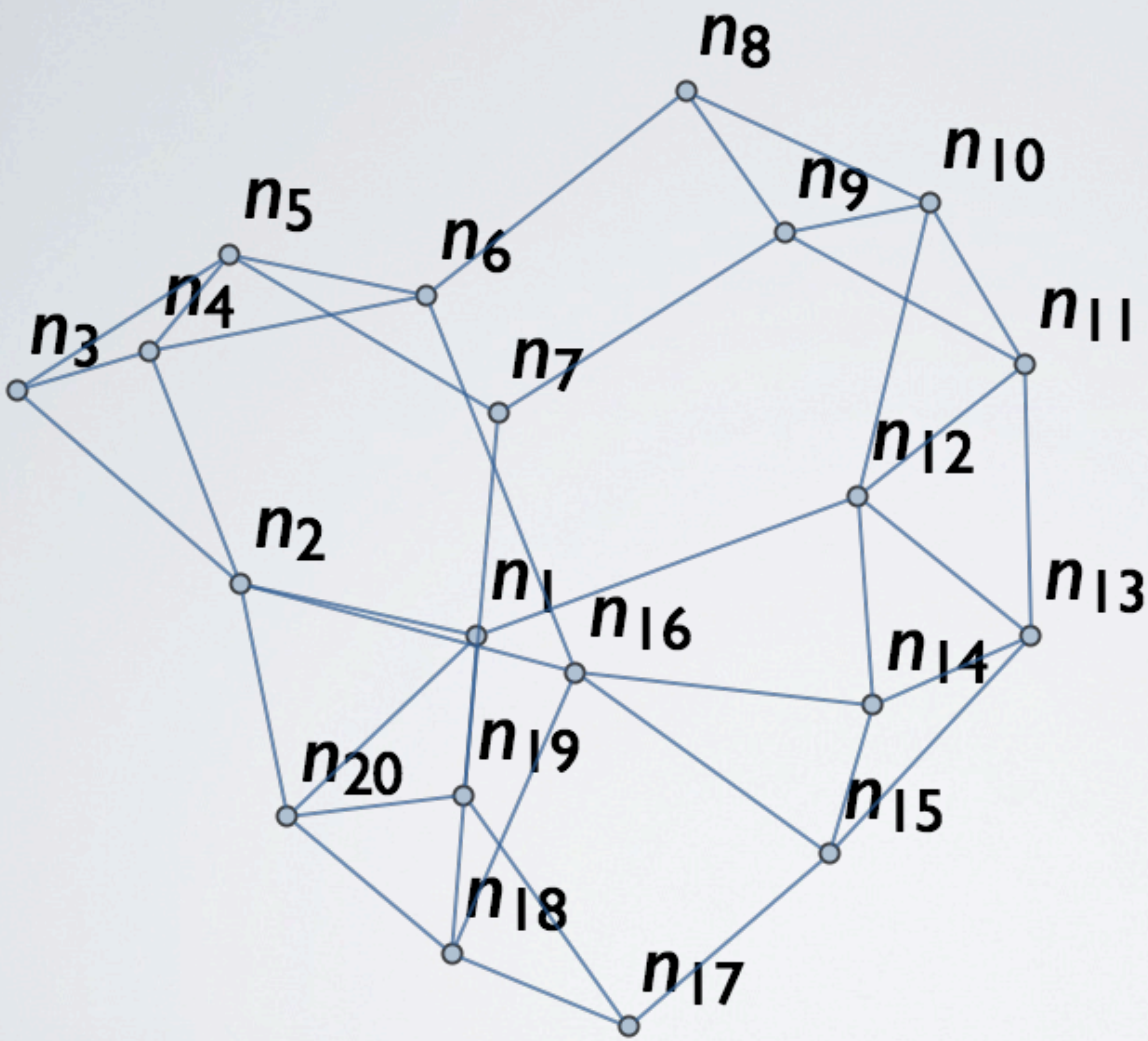
Loophole 7

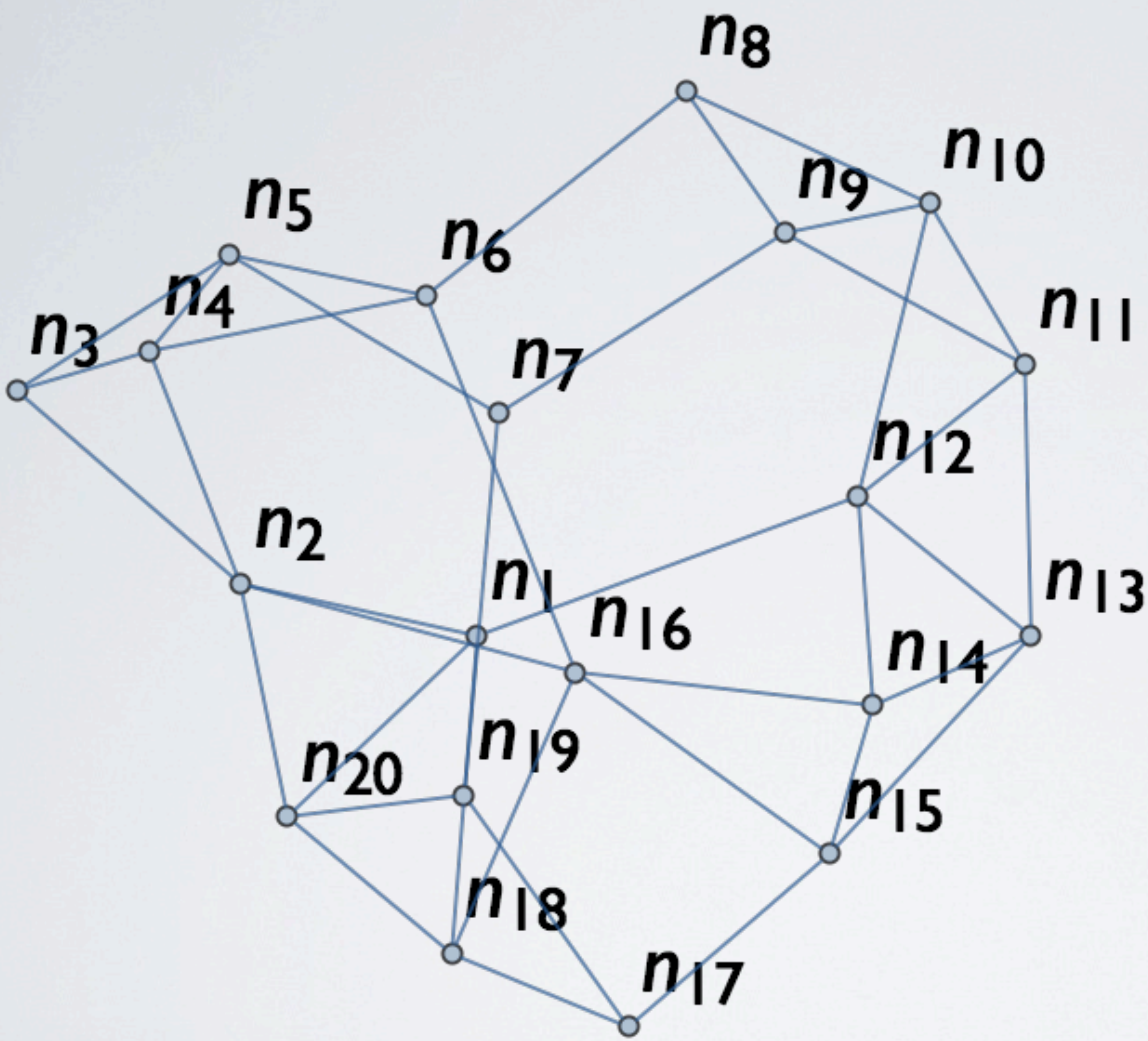
Loophole 8

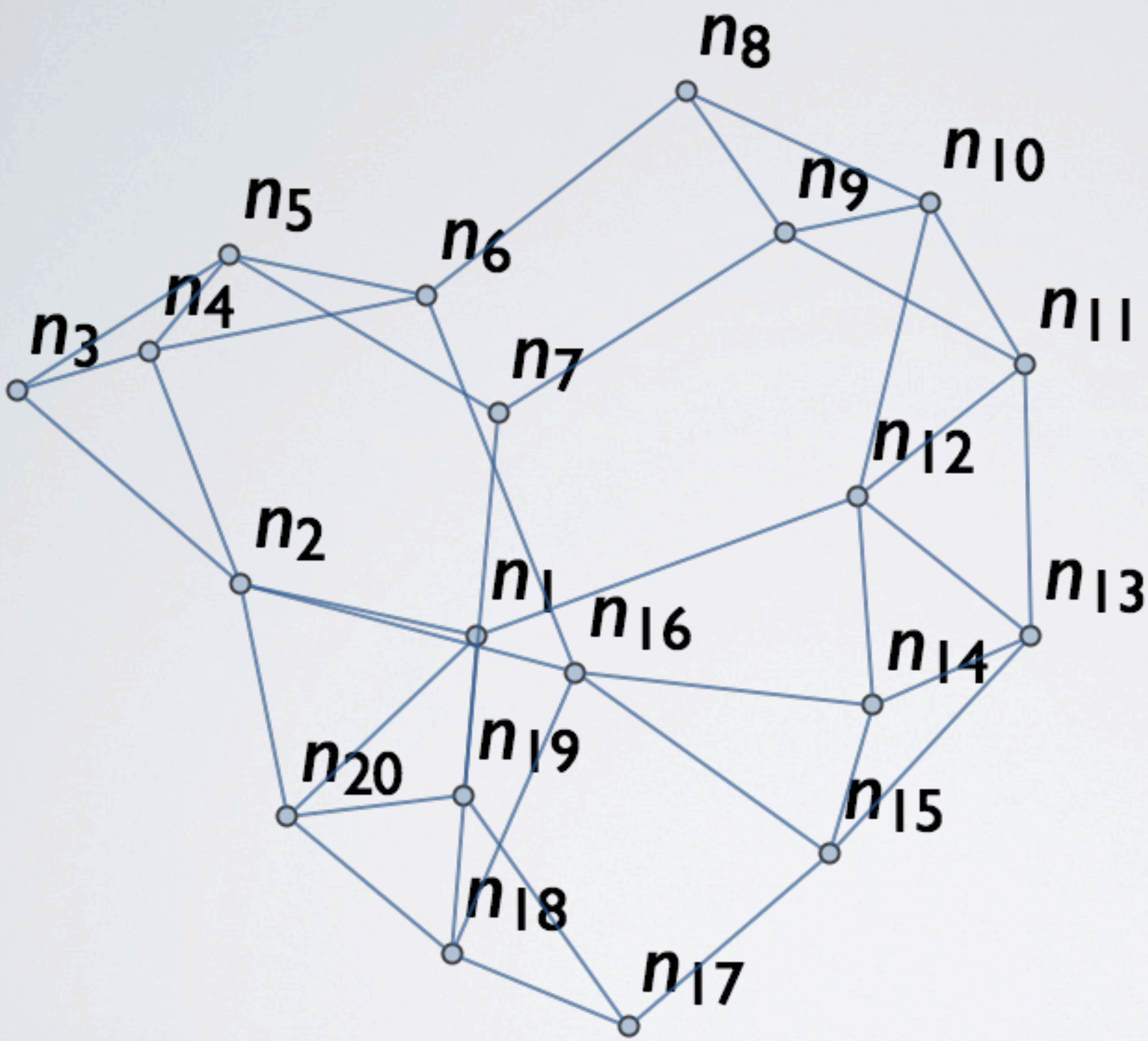
Loophole 8
Use the Force

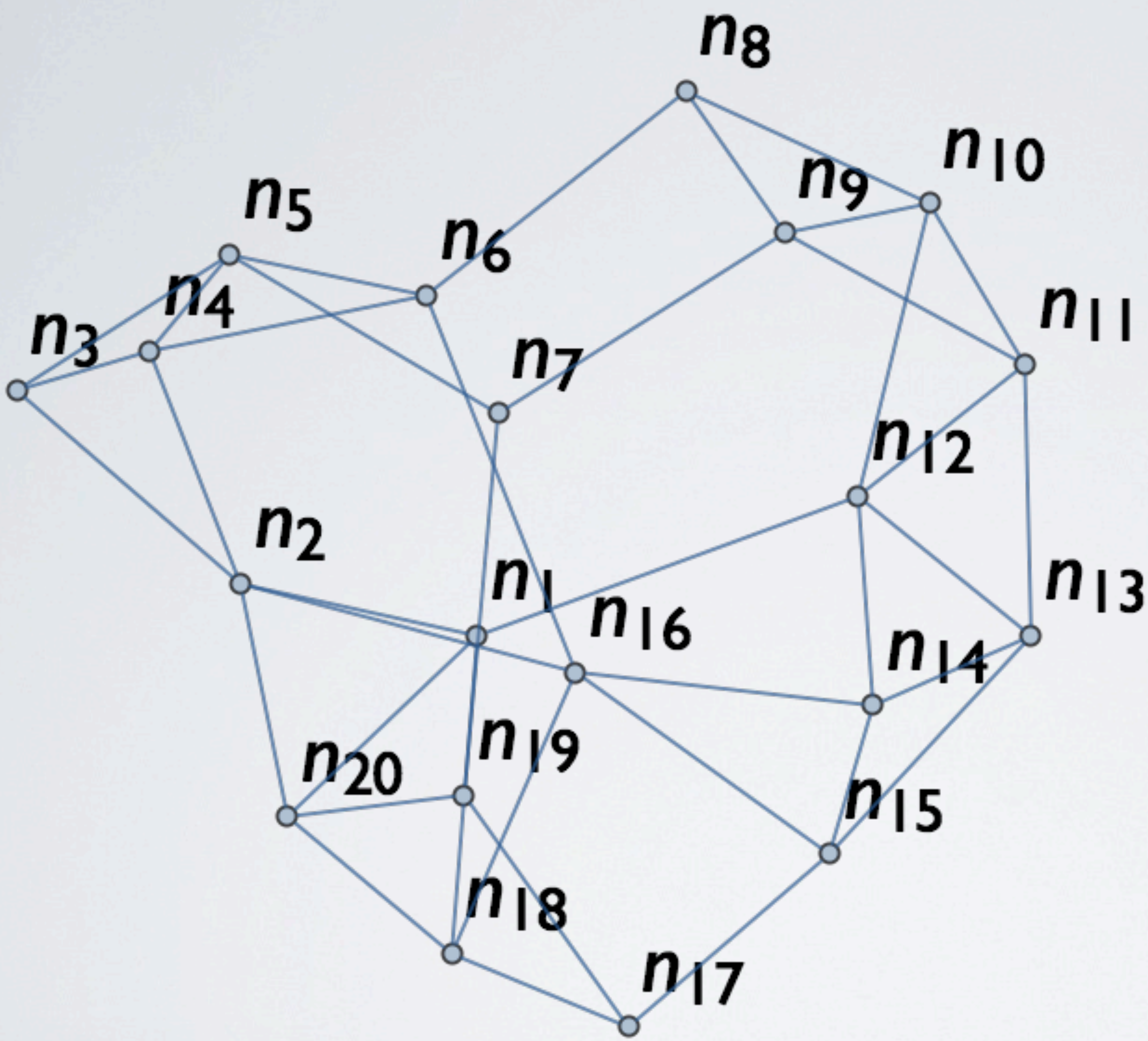
Relativistic Quantum Field Theory


n_8
n_{10}
n_9
n_5
n_6
n_4
n_{11}
n_3
n_7
n_{12}
n_2
n_1
n_{13}
n_{16}
n_{14}
n_{19}
n_{20}
n_{15}
n_{18}
n_{17}


n_8
n_{10}
n_9
n_5
n_6
n_4
n_{11}
n_3
n_7
n_{12}
n_2
n_{13}
n_1
n_{16}
n_{14}
n_{19}
n_{20}
n_{15}
n_{18}
n_{17}


n_8
n_{10}
n_9
n_5
n_6
n_4
n_{11}
n_3
n_7
n_{12}
n_2
n_{13}
n_1
n_{16}
n_{14}
n_{19}
n_{20}
n_{15}
n_{18}
n_{17}


n_8
n_{10}
n_9
n_5
n_6
n_4
n_{11}
n_3
n_7
n_{12}
n_2
n_{13}
n_1
n_{16}
n_{14}
n_{19}
n_{20}
n_{15}
n_{18}
n_{17}


n_8
n_{10}
n_9
n_5
n_6
n_4
n_{11}
n_3
n_7
n_{12}
n_2
n_1
n_{13}
n_{16}
n_{14}
n_{19}
n_{20}
n_{15}
n_{18}
n_{17}

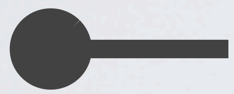
GPS

Loophole 9

Redefine availability

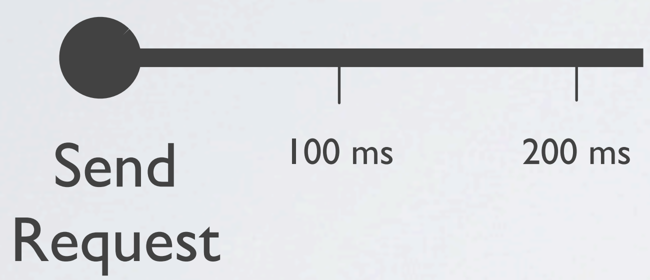
	Normal Operation	Partition Detected
Query	Available	Available
Alter	Available	Not available

ASYMMETRY OF TIME

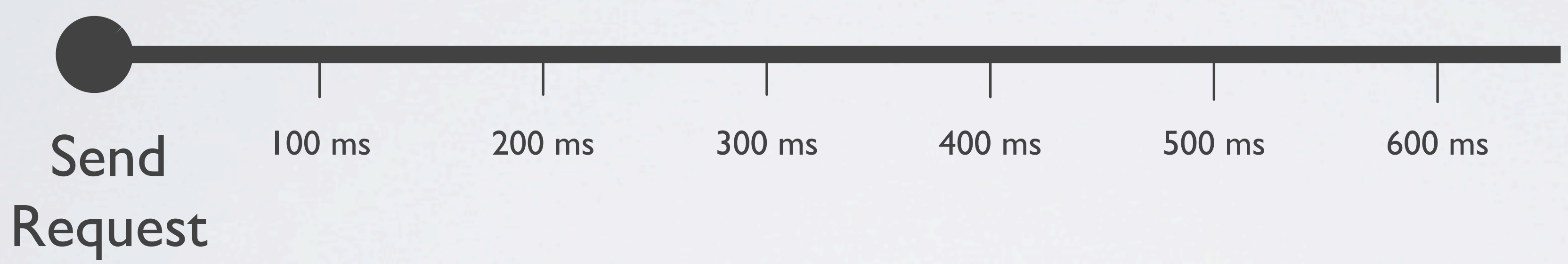


Send
Request

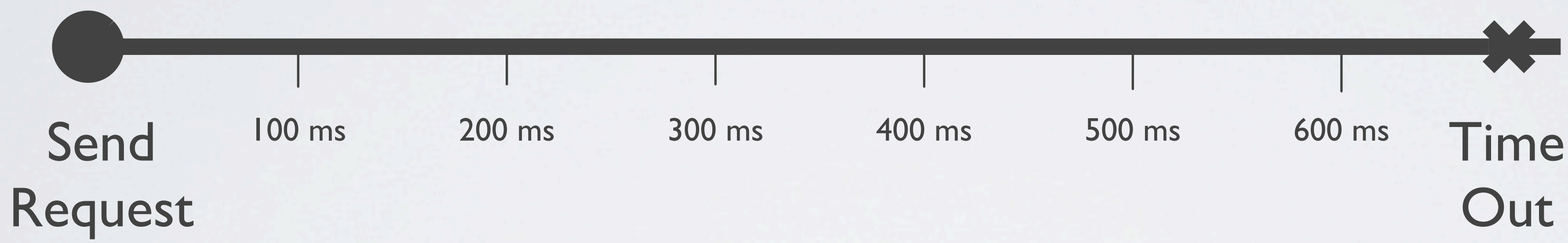
ASYMMETRY OF TIME

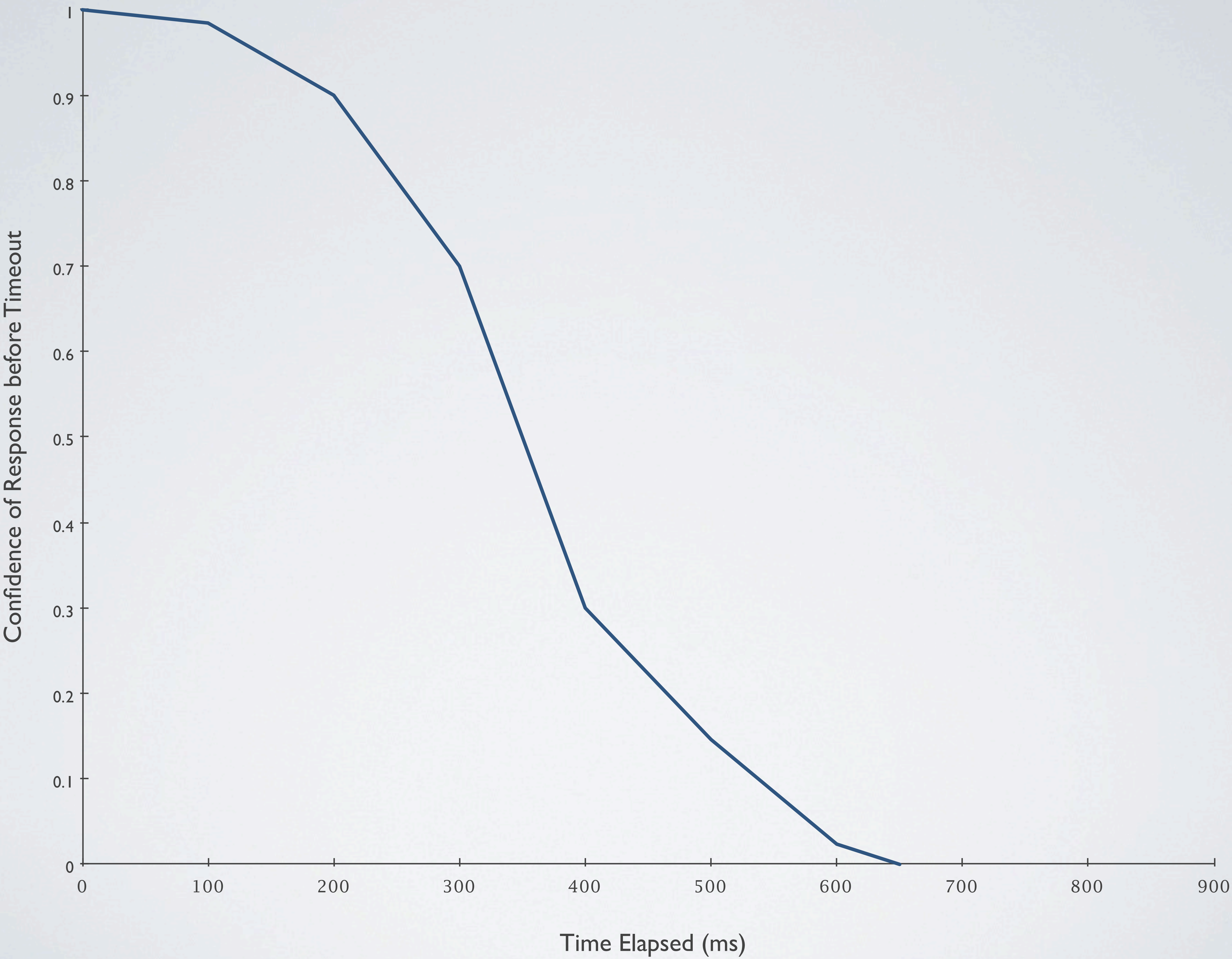


ASYMMETRY OF TIME

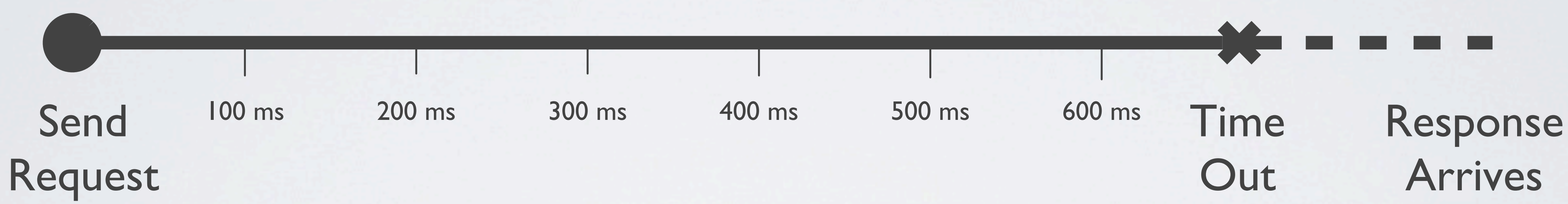


ASYMMETRY OF TIME




Confidence of Response before Timeout
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
100
200
300
400
500
600
700
800
900
Time Elapsed (ms)

ASYMMETRY OF TIME



To the observer, there is no difference between “too slow” and “not there”.

P A C E L C

P

E

A

L

C

C

Partition?

A

L

C

C

Partition?

Yes

A

L

C

C

Partition?

Yes

Availability L

vs

Consistency C

Partition?

Yes | No

Availability | L

vs

Consistency | C

Partition?

Yes

No

Availability

vs

Consistency

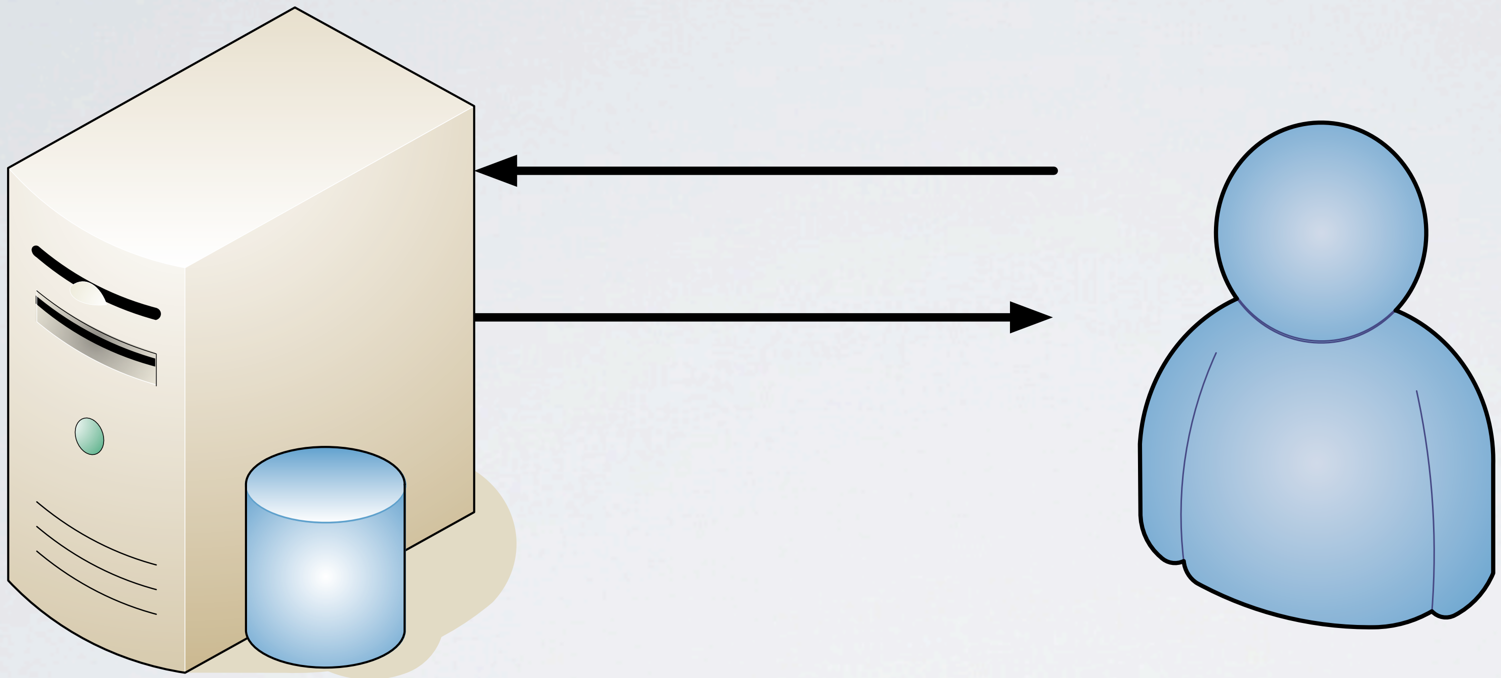
Latency

vs

Consistency

Loophole 10

OBSERVABLE CONSISTENCY

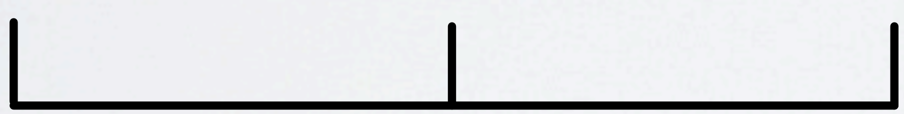


Porky Pig's Window Shade

If Porky Pig is looking at the window shade, it will be down.

If he is looking away from the window shade, it will be up.

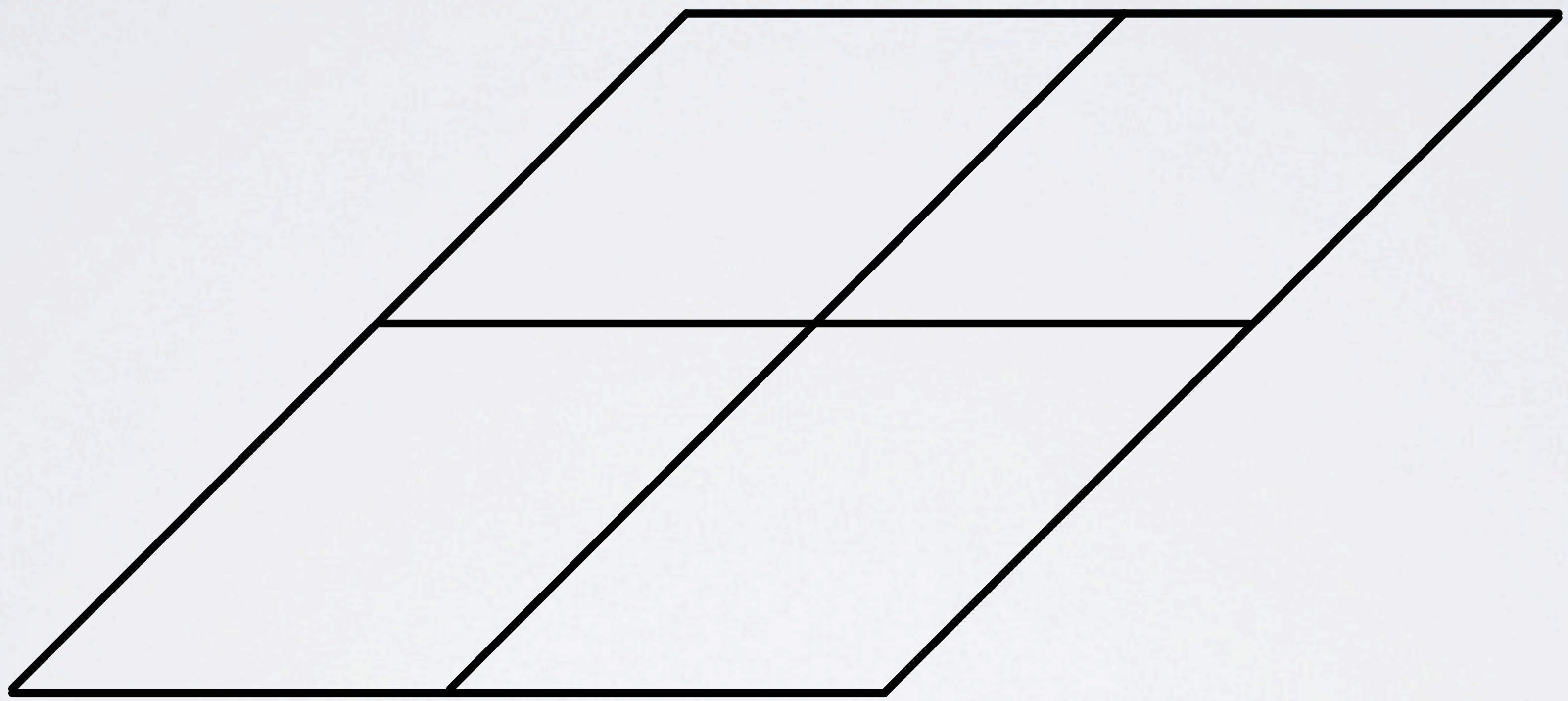
FIRST DIMENSION



X_1 = {looking, not looking}

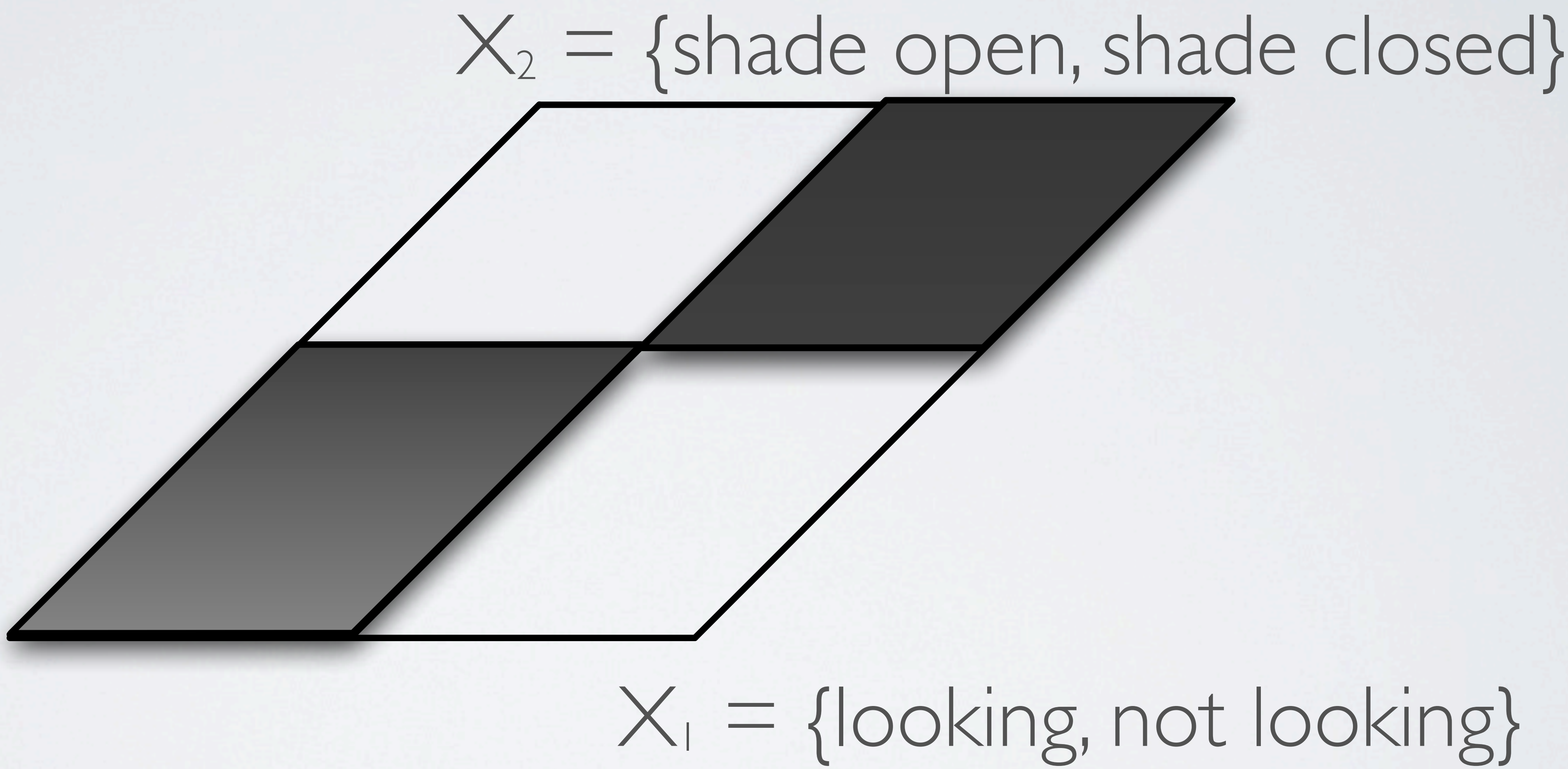
SECOND DIMENSION

X_2 = {shade open, shade closed}



X_1 = {looking, not looking}

FORBIDDEN STATES



Back to "consistency" as a predicate over the state space


time
t11
t12

Back to CAP

None of these make CAP “untrue”

None of these make CAP "untrue"

Some of them operate under different assumptions.

Some of them are **totally** impractical.

Some of them are **totally** impractical.

Some of them are in production today.

Finally, I'll close with this bit of code:

Finally, I'll close with this bit of code:

QHH9Q+++

mtnygard@thinkrelevance.com

@mtnygard

Michael T. Nygard
Relevance, Inc.