Improving Delay in a Multi-hop Wireless Network with Receiver Diversity

Tara Javidi

Electrical and Computer Engineering

University of California, San Diego

Joint Work with: Parul Gupta, M. Naghshvar, and H. Zhuang

(Acknowledgment: D. Teneketzis, C. Lott)

Opportunistic Routing: Model

Opportunistic Routing: Model

Model (M1)

single tx-type, single commodity, with orthogonal channels

[LottTeneketzis, CDC'00], [LottJTeneketzis, SN'02], [Neely, CISS'06]

Opportunistic Routing: Model

Model (M1)

single tx-type, single commodity, with orthogonal channels
[LottTeneketzis, CDC’00], [LottJTeneketzis, SN’02], [Neely, CISS’06]

- Network consists of nodes: $\{1, 2, \ldots, d\}$

Opportunistic Routing: Model

Model (M1)

single tx-type, single commodity, with orthogonal channels

[LottTeneketzis, CDC'00], [LottJTeneketzis, SN'02], [Neely, CISS'06]

- Network consists of nodes: $\{1, 2, \ldots, d\}$
- Packets are destined for d
    - $A_t(i)$: # of packets originating at node i at time t
    - Bounded and mixing random process with rate λ_i

Opportunistic Routing: Model

Model (M1)

single tx-type, single commodity, with orthogonal channels

[LottTeneketzis, CDC'00], [LottJTeneketzis, SN'02], [Neely, CISS'06]

- Network consists of nodes: $\{1, 2, \ldots, d\}$
- Packets are destined for d
  - $A_t(i)$: # of packets originating at node i at time t
  - Bounded and mixing random process with rate λ_i
- Slotted time/packet: Node i's tx of one packet takes one time slot

Opportunistic Routing: Model

Model (M1)

single tx-type, single commodity, with orthogonal channels
[LottTeneketzis, CDC'00], [LottJTeneketzis, SN'02], [Neely, CISS'06]

- Network consists of nodes: $\{1, 2, \ldots, d\}$
- Packets are destined for d
  - $A_t(i)$: # of packets originating at node i at time t
  - Bounded and mixing random process with rate λ_i
- Slotted time/packet: Node i's tx of one packet takes one time slot
- Node i's tx successfully rcved and acked by subset S of neighbors with probability $P(S|i)$ independent of other tx (orthogonal tx)

Opportunistic Routing: Model

Opportunistic Routing: Model

- Routing decision frame
  - The node responsible, i, transmits (locally broadcasts)
  - Nodes S_t successfully decode & acknowledge reception (tx outcome)
  - Actions: 1) choose a neighbor in S_t as the next relay, or 2) retransmit

Opportunistic Routing: Model

- Routing decision frame
  - The node responsible, i, transmits (locally broadcasts)
  - Nodes S_t successfully decode & acknowledge reception (tx outcome)
  - Actions: 1) choose a neighbor in S_t as the next relay, or 2) retransmit
- Routing policy maps actions to tx outcomes

Opportunistic Routing: Model

- Routing decision frame
  - The node responsible, i, transmits (locally broadcasts)
  - Nodes S_t successfully decode & acknowledge reception (tx outcome)
  - Actions: 1) choose a neighbor in S_t as the next relay, or 2) retransmit
- Routing policy maps actions to tx outcomes
- Distributed: “routing token” + three way hand-shake

Opportunistic Routing: Model

- Routing decision frame
  - The node responsible, i, transmits (locally broadcasts)
  - Nodes S_t successfully decode & acknowledge reception (tx outcome)
  - Actions: 1) choose a neighbor in S_t as the next relay, or 2) retransmit
- Routing policy maps actions to tx outcomes
- Distributed: “routing token” + three way hand-shake

Objective:

Opportunistic Routing: Model

- Routing decision frame
  - The node responsible, i, transmits (locally broadcasts)
  - Nodes S_t successfully decode & acknowledge reception (tx outcome)
  - Actions: 1) choose a neighbor in S_t as the next relay, or 2) retransmit
- Routing policy maps actions to tx outcomes
- Distributed: "routing token" + three way hand-shake

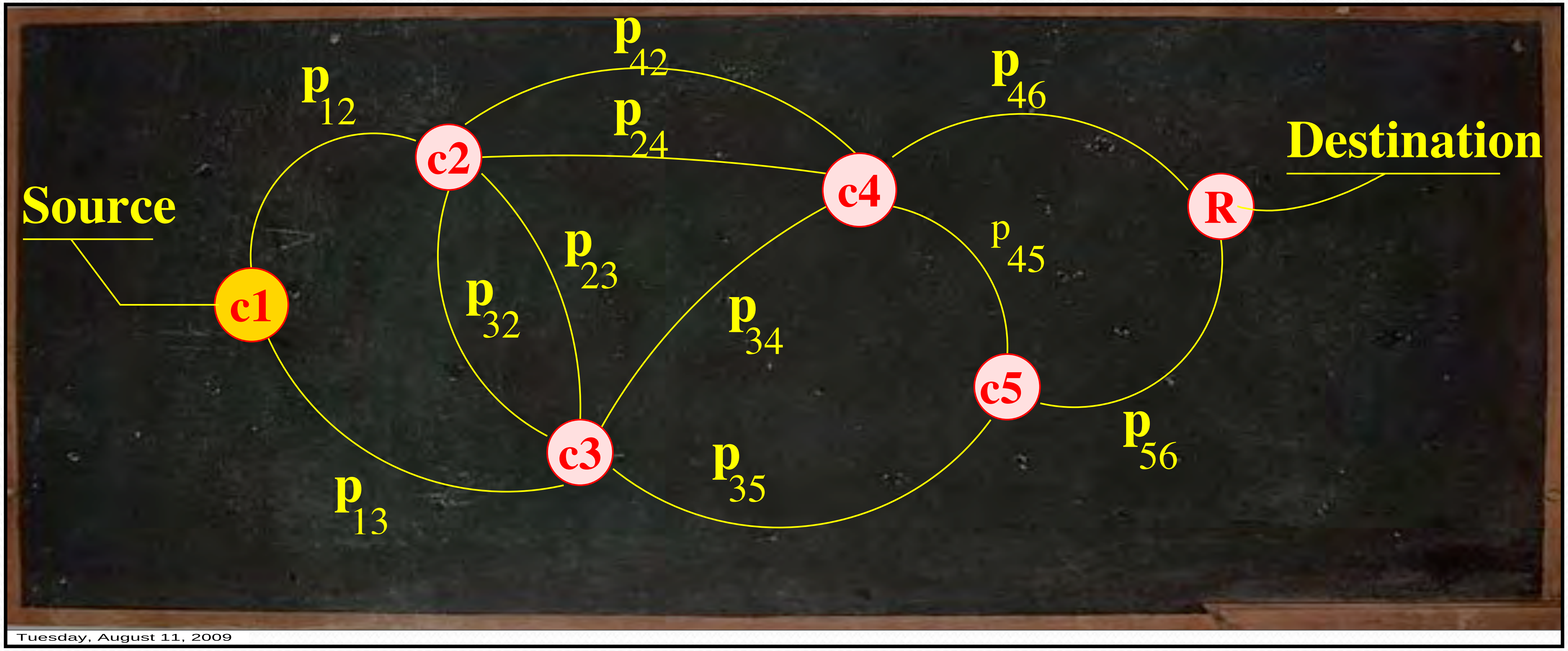
Objective:

Deliver the packets to the destination with small expected delay

- (per packet) delay ≡ interval between arrival time to delivery time

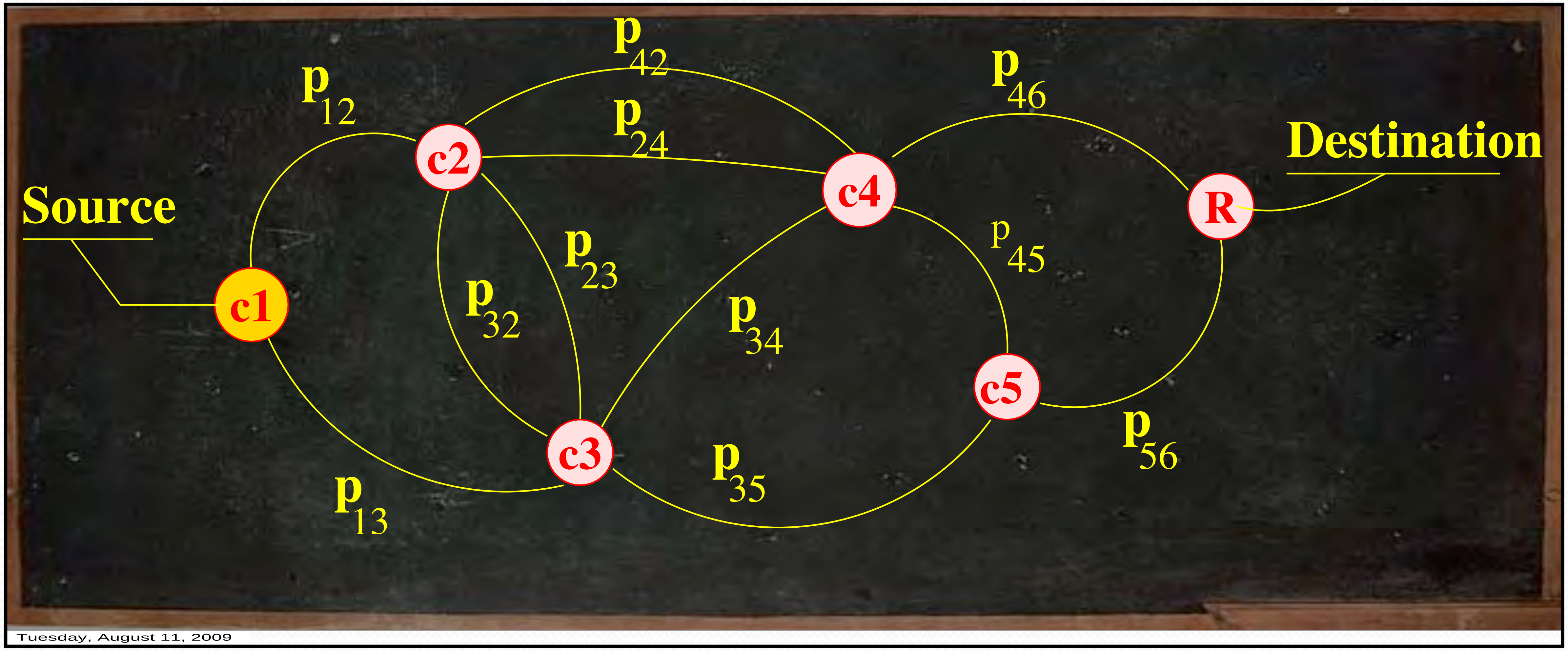
Opportunistic Routing: Model

Opportunistic Routing: Model



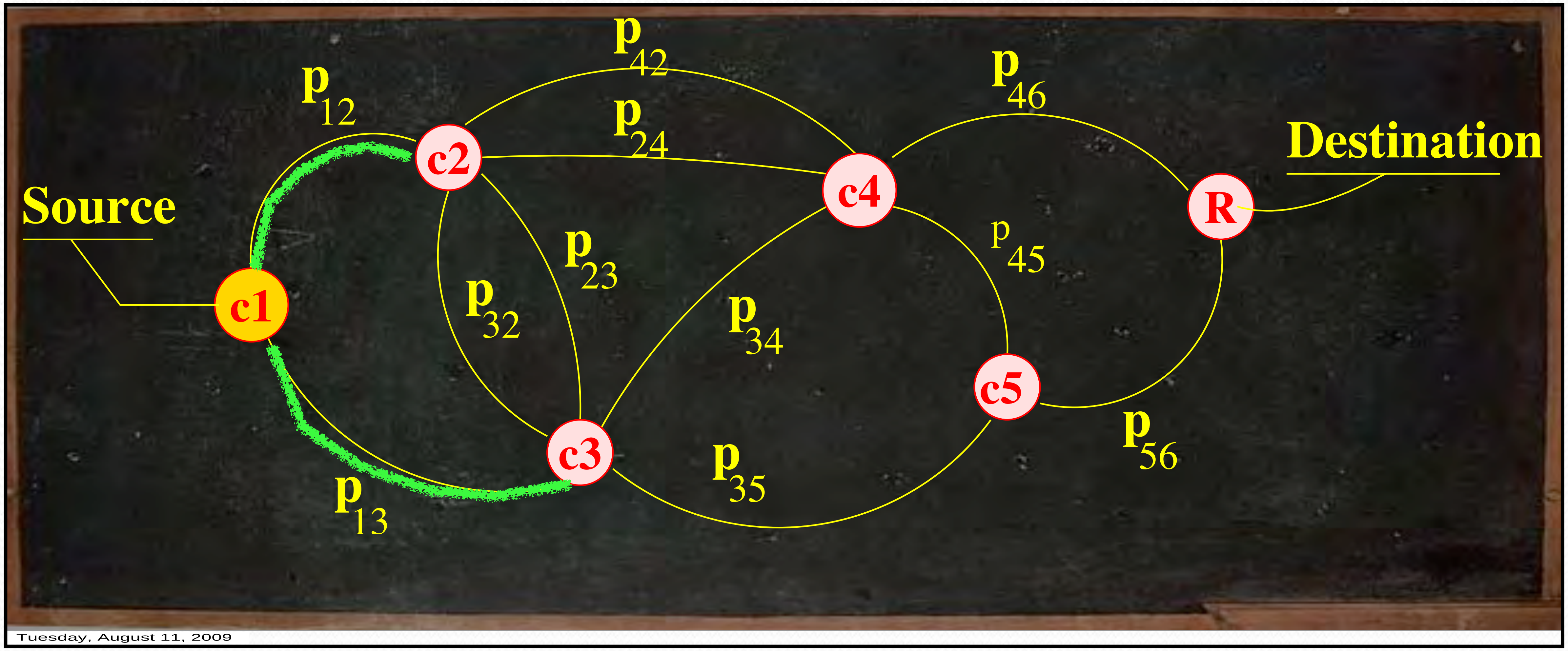
Opportunistic Routing: Model

- Transmissions received probabilistically and are acked



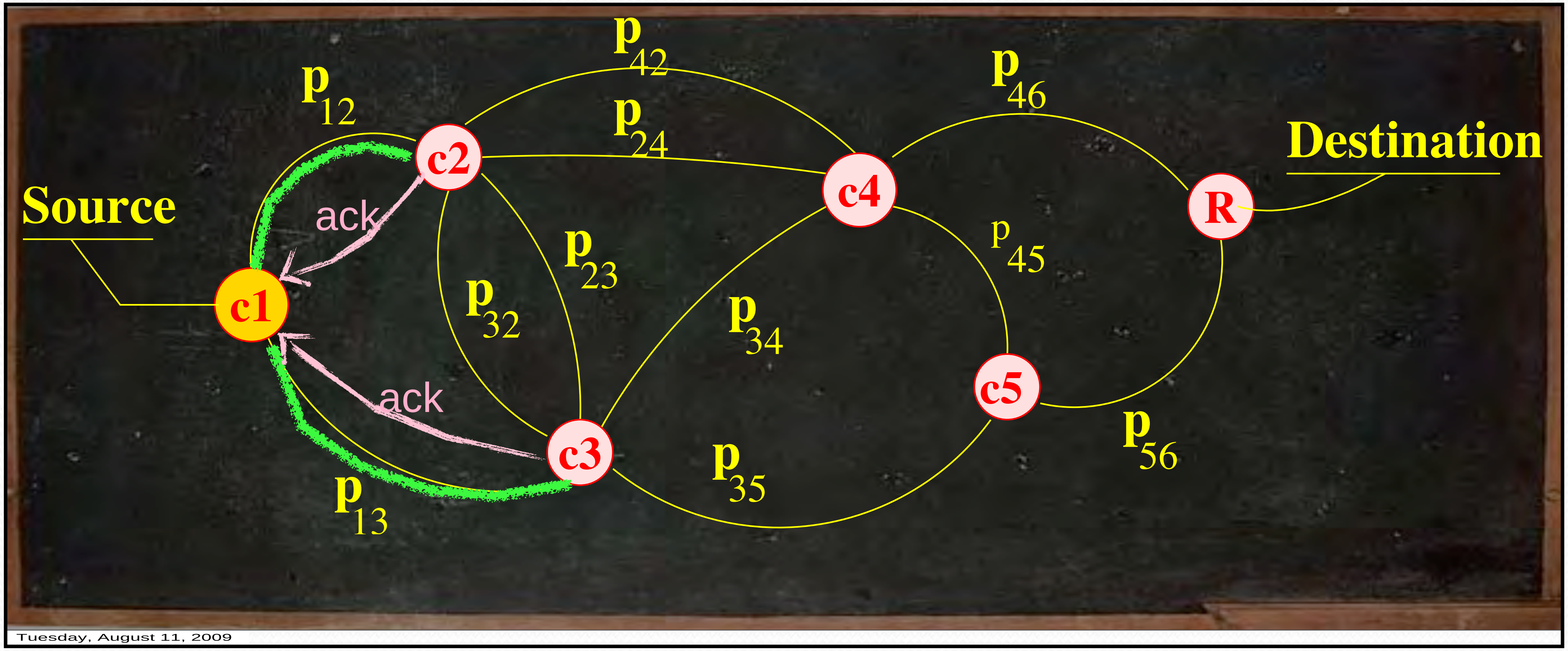
Opportunistic Routing: Model

- Transmissions received probabilistically and are acked
- The node responsible, i, transmits (locally broadcasts)



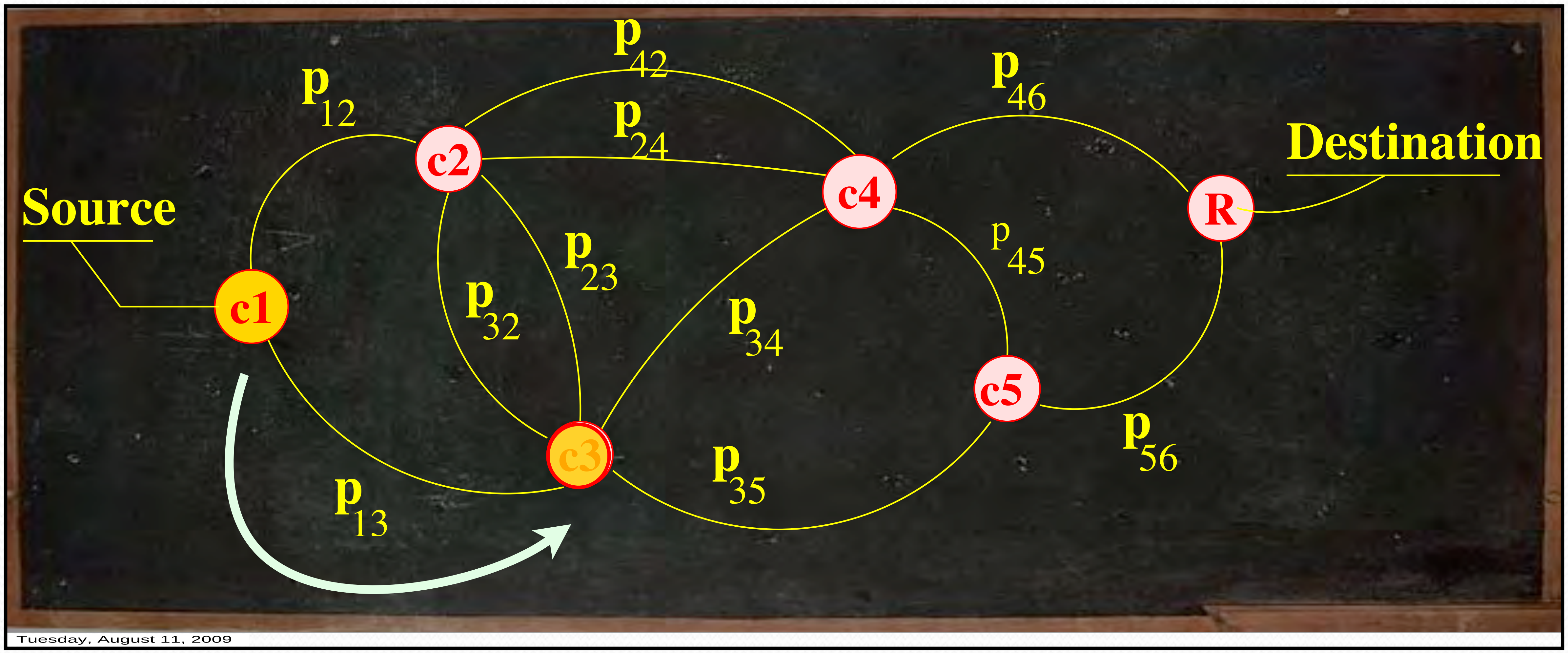
Opportunistic Routing: Model

- Transmissions received probabilistically and are acked
- The node responsible, i, transmits (locally broadcasts)
- Nodes S_t successfully decode & acknowledge reception (tx outcome)



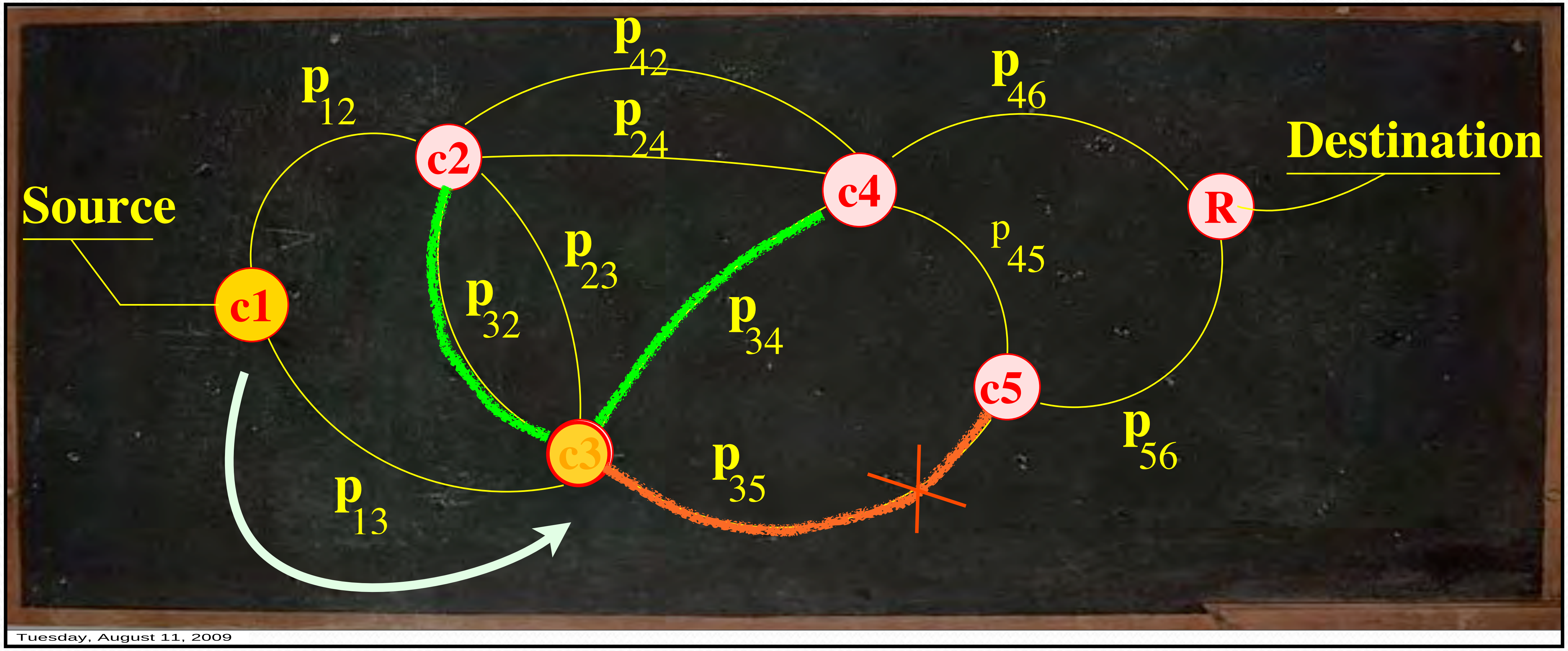
Opportunistic Routing: Model

- Transmissions received probabilistically and are acked
- The node responsible, i, transmits (locally broadcasts)
- Nodes S_t successfully decode & acknowledge reception (tx outcome)
- Actions: 1) choose a neighbor in S_t as the next relay, or 2) retransmit



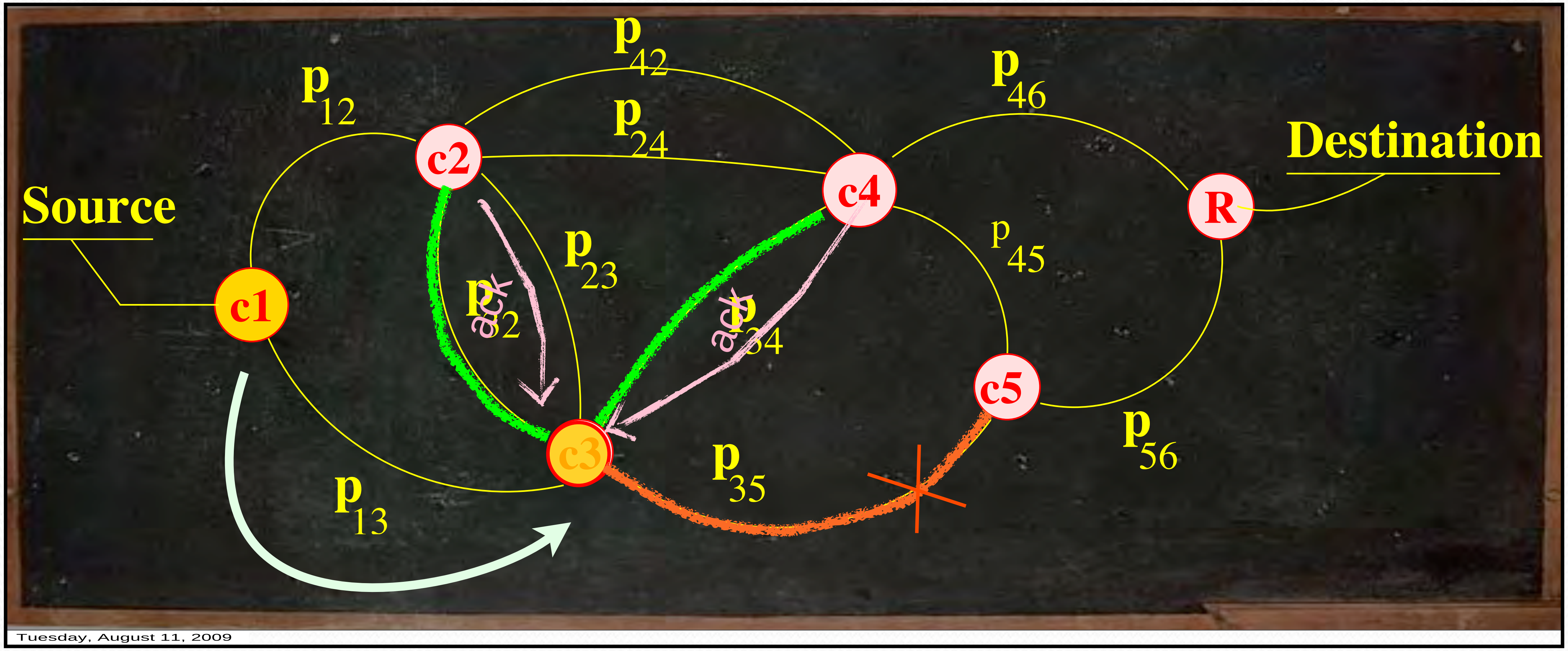
Opportunistic Routing: Model

- Transmissions received probabilistically and are acked
- The node responsible, i, transmits (locally broadcasts)
- Nodes S_t successfully decode & acknowledge reception (tx outcome)
- Actions: 1) choose a neighbor in S_t as the next relay, or 2) retransmit



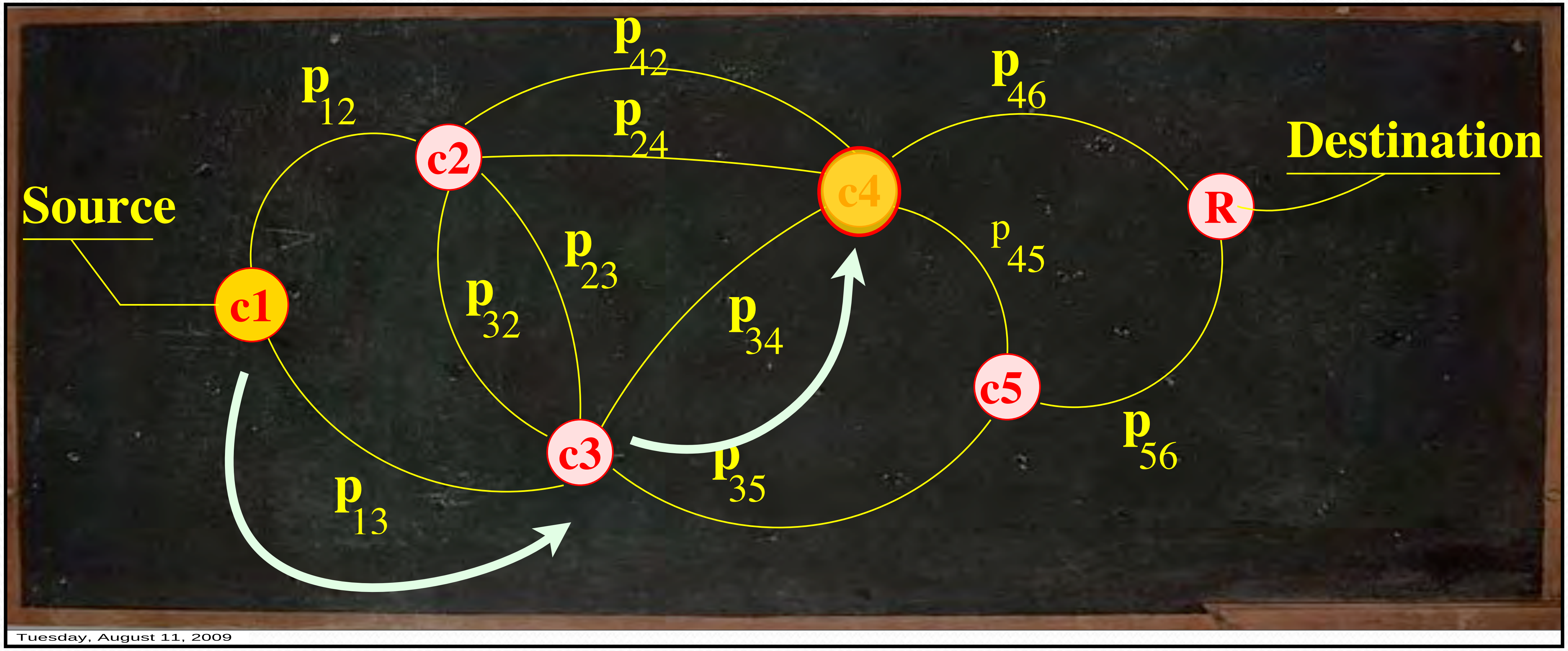
Opportunistic Routing: Model

- Transmissions received probabilistically and are acked
- The node responsible, i, transmits (locally broadcasts)
- Nodes S_t successfully decode & acknowledge reception (tx outcome)
- Actions: 1) choose a neighbor in S_t as the next relay, or 2) retransmit



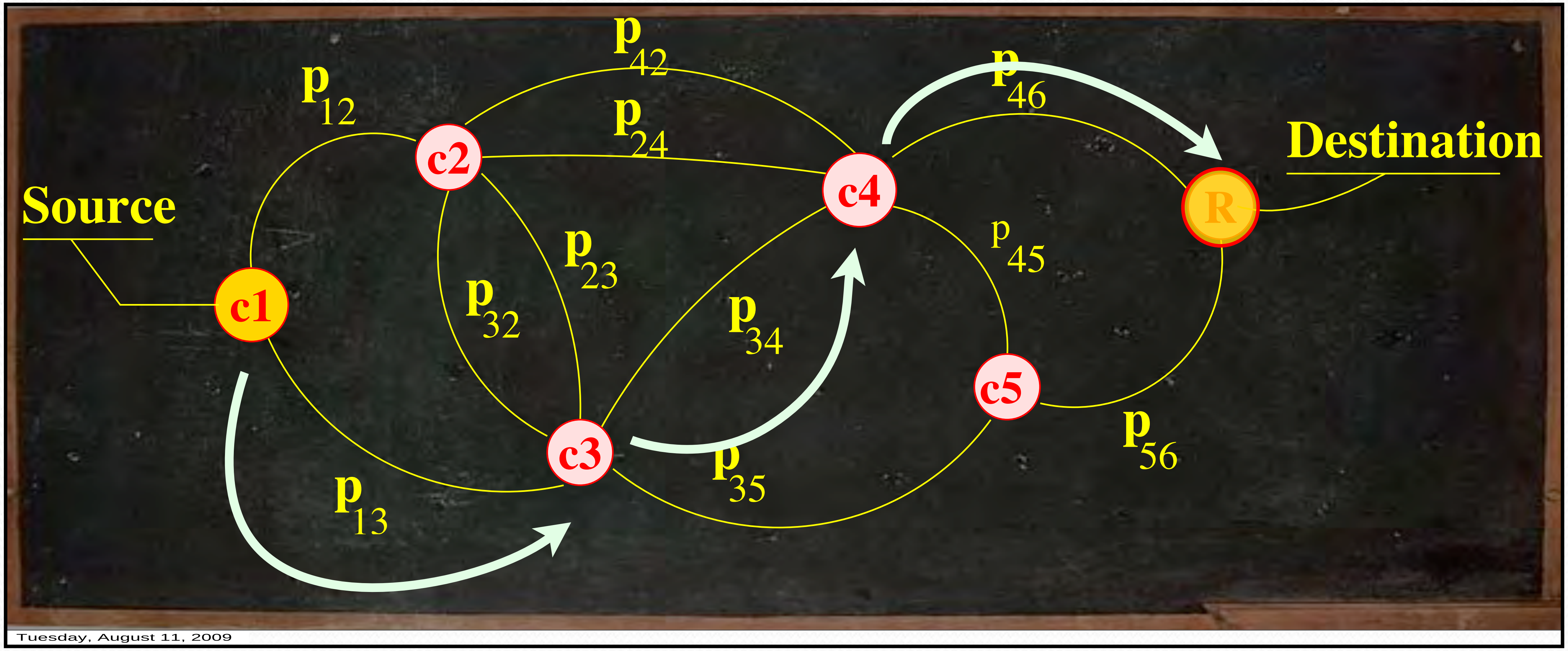
Opportunistic Routing: Model

- Transmissions received probabilistically and are acked
- The node responsible, i, transmits (locally broadcasts)
- Nodes S_t successfully decode & acknowledge reception (tx outcome)
- Actions: 1) choose a neighbor in S_t as the next relay, or 2) retransmit



Opportunistic Routing: Model

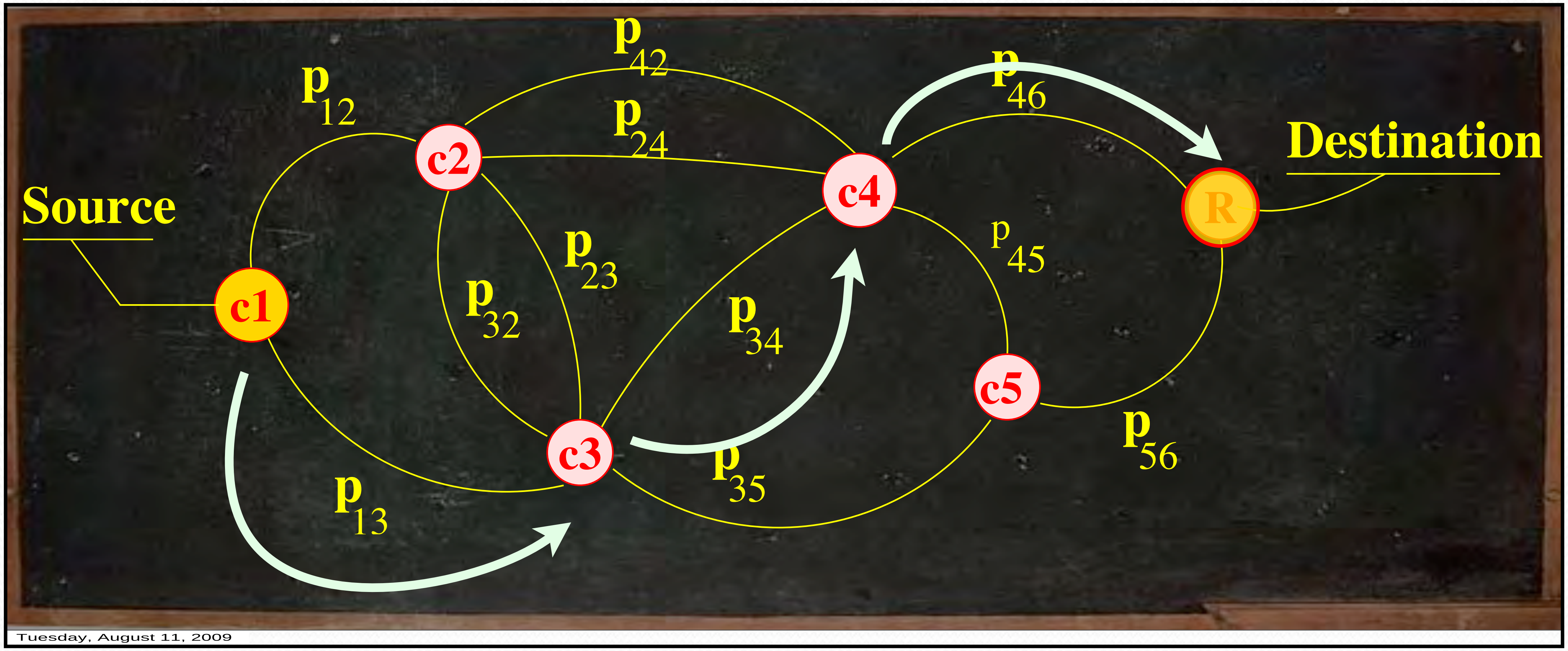
- Transmissions received probabilistically and are acked
- The node responsible, i, transmits (locally broadcasts)
- Nodes S_t successfully decode & acknowledge reception (tx outcome)
- Actions: 1) choose a neighbor in S_t as the next relay, or 2) retransmit



Opportunistic Routing: Model

- Transmissions received probabilistically and are acked

routing frame
- The node responsible, i, transmits (locally broadcasts)
- Nodes S_t successfully decode & acknowledge reception (tx outcome)
- Actions: 1) choose a neighbor in S_t as the next relay, or 2) retransmit



Monday, May 17, 2010

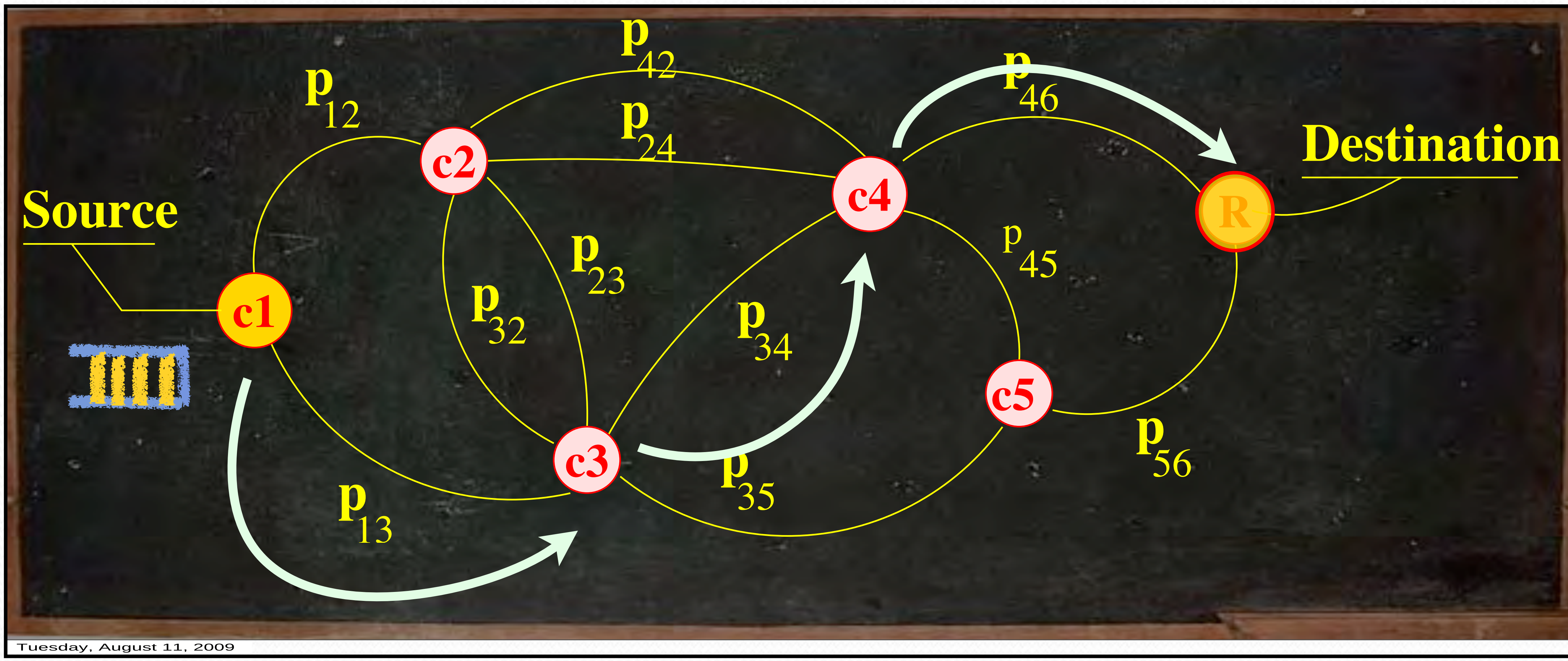
Opportunistic Routing: Model

- Transmissions received probabilistically and are acked

routing frame

- The node responsible, i, transmits (locally broadcasts)
- Nodes S_t successfully decode & acknowledge reception (tx outcome)
- Actions: 1) choose a neighbor in S_t as the next relay, or 2) retransmit

- Packets are queued up when immediate relaying not possible



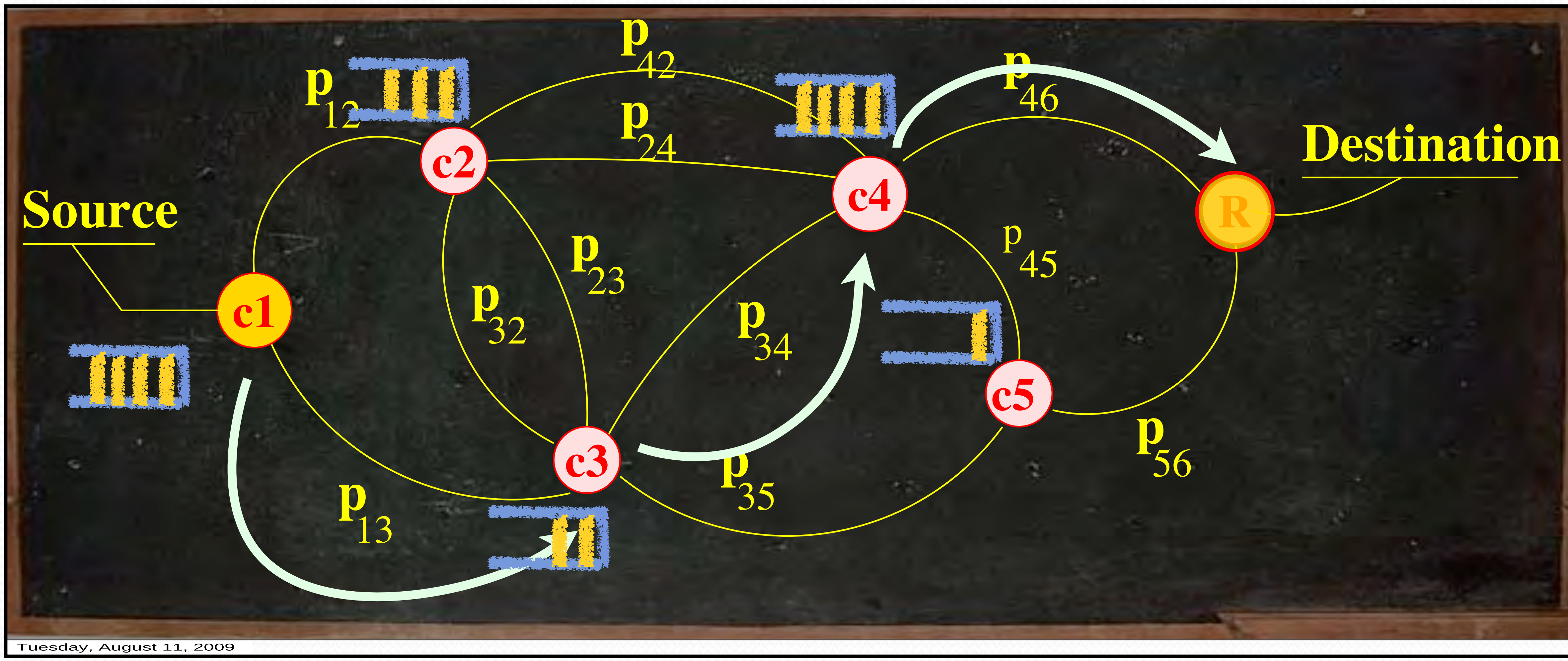
Opportunistic Routing: Model

- Transmissions received probabilistically and are acked

routing frame

- The node responsible, i, transmits (locally broadcasts)
- Nodes s_t successfully decode & acknowledge reception (tx outcome)
- Actions: 1) choose a neighbor in s_t as the next relay, or 2) retransmit

- Packets are queued up when immediate relaying not possible



Opportunistic Routing: Control Objective

Opportunistic Routing: Control Objective

- Routing determines packet departures from i to j

Opportunistic Routing: Control Objective

- Routing determines packet departures from i to j
  - $D_t(i,j)$: # of packets from i to j at time t $\left(D_t(i,j) \in \{0,1\}\right.$ and $\left.\sum_j D_t(i,j) \leq 1\right)$

Opportunistic Routing: Control Objective

- Routing determines packet departures from i to j
  - $D_t(i,j)$: # of packets from i to j at time t $\left(D_t(i,j) \in \{0,1\}\right.$ and $\left.\sum_j D_t(i,j) \leq 1\right)$
- Vector of queue backlogs: a stochastic process in

Opportunistic Routing: Control Objective

- Routing determines packet departures from i to j
  - $D_t(i,j)$: # of packets from i to j at time t ($D_t(i,j) \in \{0,1\}$ and $\sum_j D_t(i,j) \leq 1$)
- Vector of queue backlogs: a stochastic process in $\overrightarrow{q_t} \in \mathbb{Z}_+^d$

$$q_{t+1}(i) = \left[q_t(i) - \sum_j D_t(i,j) \right]^+ + A_t(i) + \sum_j D_t(j,i)$$

- Routing policy controls transitions of this process

Opportunistic Routing: Control Objective

- Routing determines packet departures from i to j
  - $D_t(i,j)$: # of packets from i to j at time t ($D_t(i,j) \in \{0,1\}$ and $\sum_j D_t(i,j) \leq 1$)
- Vector of queue backlogs: a stochastic process in $\overrightarrow{q_t} \in \mathbb{Z}_+^d$

$$q_{t+1}(i) = \left[q_t(i) - \sum_j D_t(i,j) \right]^+ + A_t(i) + \sum_j D_t(j,i)$$

- Routing policy controls transitions of this process
  - Markov under a Markov policy $\pi : \vec{q}_t \times S_t \rightarrow \vec{D}_t$

Opportunistic Routing: Control Objective

- Routing determines packet departures from i to j
  - $D_t(i,j)$: # of packets from i to j at time t ($D_t(i,j) \in \{0,1\}$ and $\sum_j D_t(i,j) \leq 1$)
- Vector of queue backlogs: a stochastic process in $\overrightarrow{q_t} \in \mathbb{Z}_+^d$

$$q_{t+1}(i) = \left[q_t(i) - \sum_j D_t(i,j) \right]^+ + A_t(i) + \sum_j D_t(j,i)$$

- Routing policy controls transitions of this process
  - Markov under a Markov policy $\pi : \vec{q}_t \times S_t \rightarrow \vec{D}_t$

Objective:

Opportunistic Routing: Control Objective

- Routing determines packet departures from i to j
  - $D_t(i,j)$: # of packets from i to j at time t ($D_t(i,j) \in \{0,1\}$ and $\sum_j D_t(i,j) \leq 1$)
- Vector of queue backlogs: a stochastic process in $\vec{q}_t \in \mathbb{Z}_+^d$

$$q_{t+1}(i) = \left[q_t(i) - \sum_j D_t(i,j) \right]^+ + A_t(i) + \sum_j D_t(j,i)$$

- Routing policy controls transitions of this process
  - Markov under a Markov policy $\pi : \vec{q}_t \times S_t \to \vec{D}_t$

Objective:

Find policy with small $\mathbb{E}\left\{ \sum_i q_{t+1}(i) \right\}$

Opportunistic Routing: Control Objective

- Routing determines packet departures from i to j
  - $D_t(i,j)$: # of packets from i to j at time t ($D_t(i,j) \in \{0,1\}$ and $\sum_j D_t(i,j) \le 1$)
- Vector of queue backlogs: a stochastic process in $\overrightarrow{q_t} \in \mathbb{Z}_+^d$

$$q_{t+1}(i) = \left[q_t(i) - \sum_j D_t(i,j) \right]^+ + A_t(i) + \sum_j D_t(j,i)$$

- Routing policy controls transitions of this process
  - Markov under a Markov policy $\pi : \vec{q}_t \times S_t \to \vec{D}_t$

Objective:

Find policy with small $\mathbb{E}\left\{ \sum_i q_{t+1}(i) \right\}$ ($\overset{\text{Little's Law}}{\Longrightarrow}$ small mean delay)

Opportunistic Routing: Control Objective

- Routing determines packet departures from i to j
  - $D_t(i,j)$: # of packets from i to j at time t ($D_t(i,j) \in \{0,1\}$ and $\sum_j D_t(i,j) \leq 1$)
- Vector of queue backlogs: a stochastic process in $\overrightarrow{q_t} \in \mathbb{Z}_+^d$

$$q_{t+1}(i) = \left[q_t(i) - \sum_j D_t(i,j) \right]^+ + A_t(i) + \sum_j D_t(j,i)$$

- Routing policy controls transitions of this process
  - Markov under a Markov policy $\pi : \vec{q}_t \times S_t \to \vec{D}_t$

Objective:

Find policy with small $\mathbb{E}\left\{ \sum_i q_{t+1}(i) \right\}$ ($\overset{\text{Little's Law}}{\Longrightarrow}$ small mean delay)

when ($\lambda_1, \ldots, \lambda_{d-1}$) admissible: at least one policy with finite delay

Opportunistic Routing: What is known?

Opportunistic Routing: What is known?

- Routing Policy $\Leftrightarrow$ rank ordering used to make routing decisions

Opportunistic Routing: What is known?

- Routing Policy $\Leftrightarrow$ rank ordering used to make routing decisions
- Opportunistic Shortest Path Routing [LT'00]

Opportunistic Routing: What is known?

- Routing Policy $\Leftrightarrow$ rank ordering used to make routing decisions
- Opportunistic Shortest Path Routing [LT'00]
  - Rank order nodes based on the expected transmissions (ETX, distance)

Opportunistic Routing: What is known?

- Routing Policy $\Leftrightarrow$ rank ordering used to make routing decisions
- Opportunistic Shortest Path Routing [LT'00]
  - Rank order nodes based on the expected transmissions (ETX, distance)
  - **Optimal** if <u>one packet</u> in network, but unbounded delay in high traffic

Opportunistic Routing: What is known?

- Routing Policy $\Leftrightarrow$ rank ordering used to make routing decisions
- Opportunistic Shortest Path Routing [LT'00]
  - Rank order nodes based on the expected transmissions (ETX, distance)
  - **Optimal** if <u>one packet</u> in network, but unbounded delay in high traffic
- Diversity Backpressure Routing [N'06]

Opportunistic Routing: What is known?

- Routing Policy $\Leftrightarrow$ rank ordering used to make routing decisions
- Opportunistic Shortest Path Routing [LT'00]
  - Rank order nodes based on the expected transmissions (ETX, distance)
  - **Optimal** if <u>one packet</u> in network, but unbounded delay in high traffic
- Diversity Backpressure Routing [N'06]
  - Rank orders nodes based on their current queue backlog

Opportunistic Routing: What is known?

- Routing Policy $\Leftrightarrow$ rank ordering used to make routing decisions
- Opportunistic Shortest Path Routing [LT'00]
  - Rank order nodes based on the expected transmissions (ETX, distance)
  - **Optimal** if <u>one packet</u> in network, but unbounded delay in high traffic
- Diversity Backpressure Routing [N'06]
  - Rank orders nodes based on their current queue backlog
  - **Throughput optimal** (bounded delay) but large delay in <u>low</u> traffic

Opportunistic Routing: What is known?

- Routing Policy $\Leftrightarrow$ rank ordering used to make routing decisions
- Opportunistic Shortest Path Routing [LT'00]
  - Rank order nodes based on the expected transmissions (ETX, distance)
  - **Optimal** if one packet in network, but unbounded delay in high traffic
- Diversity Backpressure Routing [N'06]
  - Rank orders nodes based on their current queue backlog
  - **Throughput optimal** (bounded delay) but large delay in low traffic

... and some "unsuccessful" heuristics doing both [N'07] [YSR'09]

Opportunistic Routing: What is known?

- Routing Policy ⇔ rank ordering used to make routing decisions
- Opportunistic Shortest Path Routing [LT'00]
  - Rank order nodes based on the expected transmissions (ETX, distance)
  - **Optimal** if <u>one packet</u> in network, but unbounded delay in high traffic
- Diversity Backpressure Routing [N'06]
  - Rank orders nodes based on their current queue backlog
  - **Throughput optimal** (bounded delay) but large delay in <u>low</u> traffic

... and some "unsuccessful" heuristics doing both [N'07] [YSR'09]

e.g. rank ordering based on the sum of ETX and backlog

Objective and Contributions (outline)

Objective and Contributions (outline)

Design of an opportunistic routing policy which **exhibits good delay performance** in all traffic conditions

Objective and Contributions (outline)

Design of an opportunistic routing policy which **exhibits good delay performance** in all traffic conditions

→ **Opportunistic Routing with Congestion Diversity (ORCD)**

Objective and Contributions (outline)

Design of an opportunistic routing policy which **exhibits good delay performance** in all traffic conditions

Opportunistic Routing with Congestion Diversity (ORCD)

- Nodes are ordered based on approximate expected delivery time

Objective and Contributions (outline)

Design of an opportunistic routing policy which **exhibits good delay performance** in all traffic conditions

Opportunistic Routing with Congestion Diversity (ORCD)

- Nodes are ordered based on approximate expected delivery time

$$V_t(k) < V_t(j) \Longleftrightarrow k >_{\pi_i} j$$

Objective and Contributions (outline)

Design of an opportunistic routing policy which **exhibits good delay performance** in all traffic conditions

Opportunistic Routing with Congestion Diversity (ORCD)

- Nodes are ordered based on approximate expected delivery time

$$V_t(k) < V_t(j) \Longleftrightarrow k >_{\pi_i} j$$

Fact: ORCD is throughput optimal.

[Naghshvar, Zhuang, Javidi 09]

Objective and Contributions (outline)

Design of an opportunistic routing policy which **exhibits good delay performance** in all traffic conditions

Opportunistic Routing with Congestion Diversity (ORCD)

- Nodes are ordered based on approximate expected delivery time

$$V_t(k) < V_t(j) \Longleftrightarrow k >_{\pi_i} j$$

Fact: ORCD is throughput optimal.

[Naghshvar, Zhuang, Javidi 09]

- Implementing ORCD requires extensive control overhead

Objective and Contributions (outline)

Design of an opportunistic routing policy which **exhibits good delay performance** in all traffic conditions

Opportunistic Routing with Congestion Diversity (ORCD)

- Nodes are ordered based on approximate expected delivery time

$$V_t(k) < V_t(j) \Longleftrightarrow k >_{\pi_i} j$$

Fact: ORCD is throughput optimal.

[Naghshvar, Zhuang, Javidi 09]

- Implementing ORCD requires extensive control overhead
  - Modifications of ORCD with lower overhead
  - Performance evaluation using simulations

Computation of $V_i(t)$'s

Computation of $V_i(t)$'s

Congestion measures $V_i(t)$ are solutions to the following fixed point equation:

$$\begin{cases} V_D(t) = 0, \\ V_i(t) = \mathrm{Q}_i(t) + \sum_{\mathrm{S} \subseteq \Omega} P(\mathrm{S} \mid i) \min_{j \in \mathrm{S}} V_j(t) \end{cases}$$

Computation of $V_i(t)$'s

Congestion measures $V_i(t)$ are solutions to the following fixed point equation:

$$\begin{cases} V_D(t) = 0, \\ V_i(t) = Q_i(t) + \sum_{S \subseteq \Omega} P(S \mid i) \min_{j \in S} V_j(t) \end{cases}$$

- $Q_i(t)$: backlog of node *i* at time *t*

Computation of $V_i(t)$'s

Congestion measures $V_i(t)$ are solutions to the following fixed point equation:

$$\begin{cases} V_D(t) = 0, \\ V_i(t) = \mathrm{Q}_i(t) + \sum_{\mathrm{S} \subseteq \Omega} P(\mathrm{S} \mid i) \min_{j \in \mathrm{S}} V_j(t) \end{cases}$$

- $\mathrm{Q}_i(t)$: backlog of node i at time t

Computation of $V_i(t)$'s

Congestion measures $V_i(t)$ are solutions to the following fixed point equation:

$$\begin{cases} V_D(t) = 0, \\ V_i(t) = \mathrm{Q}_i(t) + \sum_{\mathrm{S} \subseteq \Omega} P(\mathrm{S} \mid i) \min_{j \in \mathrm{S}} V_j(t) \end{cases}$$

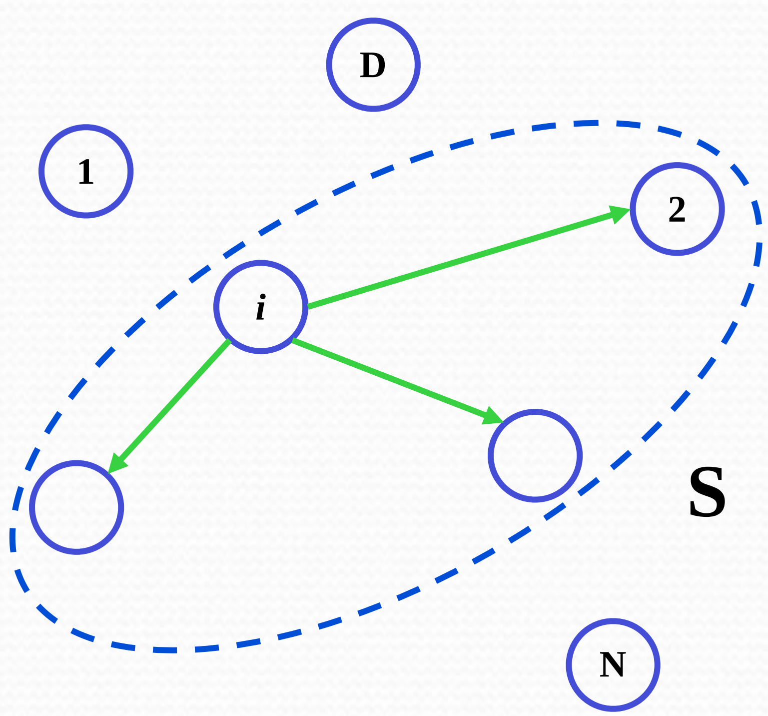
- $\mathrm{Q}_i(t)$: backlog of node i at time t

Computation of $V_i(t)$'s

Congestion measures $V_i(t)$ are solutions to the following fixed point equation:

$$\begin{cases} V_D(t) = 0, \\ V_i(t) = Q_i(t) + \sum_{S \subseteq \Omega} P(S \mid i) \min_{j \in S} V_j(t) \end{cases}$$

- $Q_i(t)$: backlog of node i at time t
- $P(S|i)$: probability that **S** be the set of nodes that receive the packet transmitted by node i

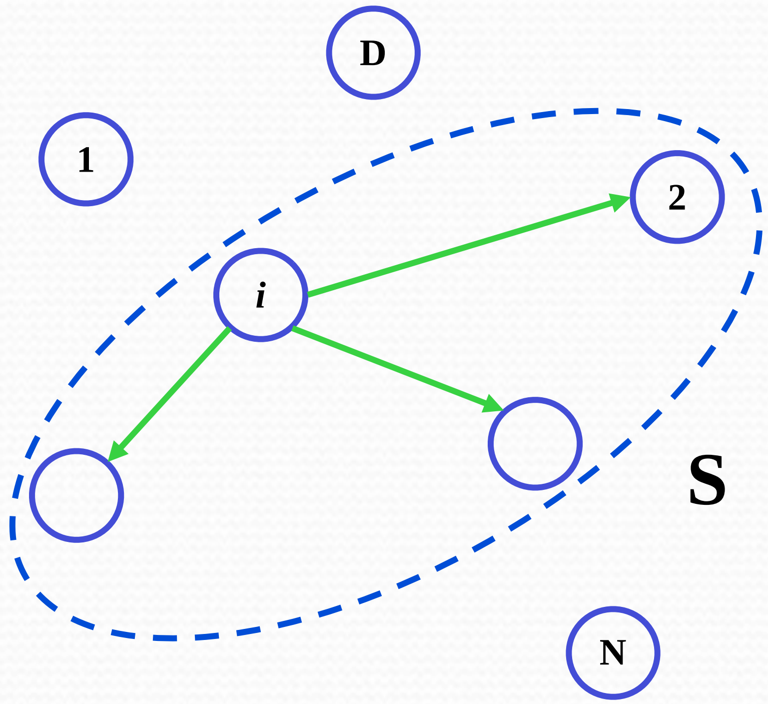


Computation of $V_i(t)$'s

Congestion measures $V_i(t)$ are solutions to the following fixed point equation:

$$\begin{cases} V_D(t) = 0, \\ V_i(t) = \mathrm{Q}_i(t) + \sum_{\mathrm{S} \subseteq \Omega} P(\mathrm{S} \mid i) \min_{j \in \mathrm{S}} V_j(t) \end{cases}$$

- $\mathrm{Q}_i(t)$: backlog of node i at time t
- $P(\mathrm{S}|i)$: probability that **S** be the set of nodes that receive the packet transmitted by node i

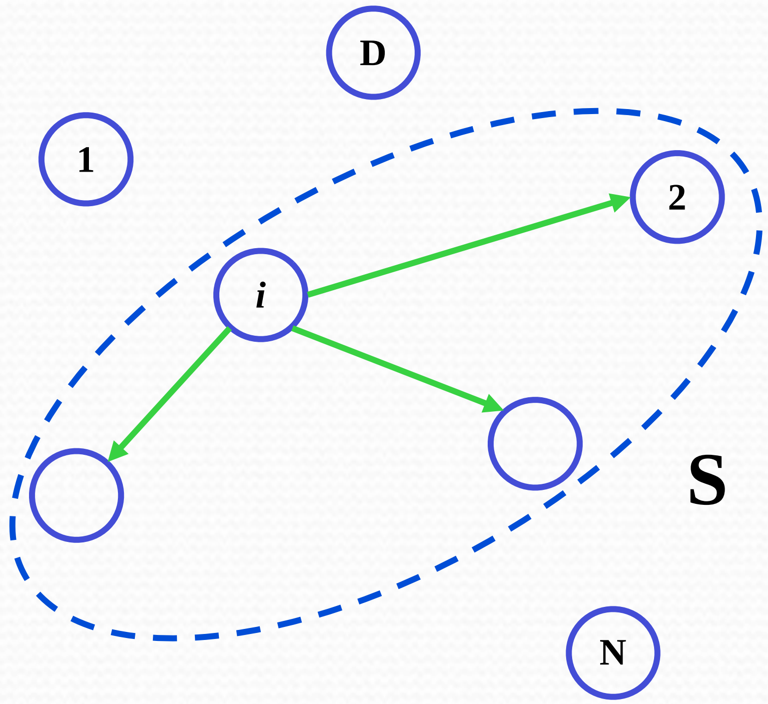


Computation of $V_i(t)$'s

Congestion measures $V_i(t)$ are solutions to the following fixed point equation:

$$\begin{cases} V_D(t) = 0, \\ V_i(t) = Q_i(t) + \sum_{S \subseteq \Omega} P(S \mid i) \min_{j \in S} V_j(t) \end{cases}$$

- $Q_i(t)$: backlog of node *i* at time *t*
- $P(S|i)$: probability that **S** be the set of nodes that receive the packet transmitted by node *i*

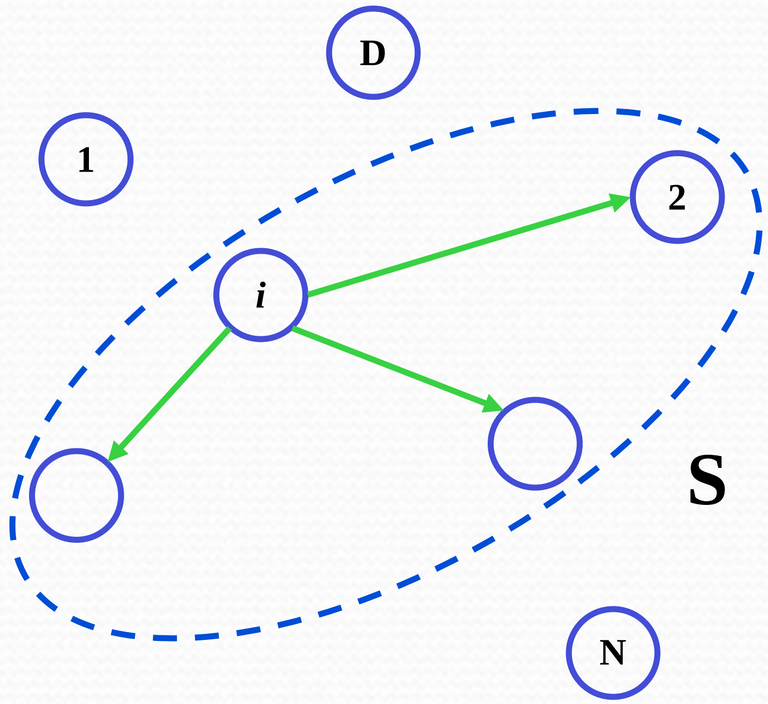


Computation of $V_i(t)$'s

Congestion measures $V_i(t)$ are solutions to the following fixed point equation:

$$\begin{cases} V_D(t) = 0, \\ V_i(t) = \mathrm{Q}_i(t) + \sum_{\mathrm{S} \subseteq \Omega} P(\mathrm{S} \mid i) \min_{j \in \mathrm{S}} V_j(t) \end{cases}$$

- $\mathrm{Q}_i(t)$: backlog of node i at time t
- $P(\mathrm{S}|i)$: probability that **S** be the set of nodes that receive the packet transmitted by node i
- $V_j(t)$: delivery time of node j

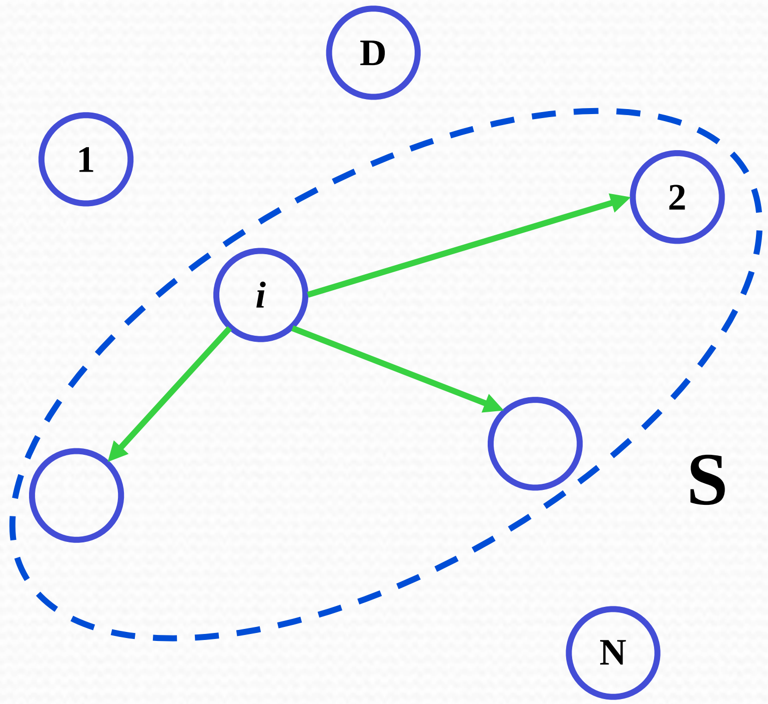


Computation of $V_i(t)$'s

Congestion measures $V_i(t)$ are solutions to the following fixed point equation:

$$\begin{cases} V_D(t) = 0, \\ V_i(t) = \mathrm{Q}_i(t) + \sum_{\mathrm{S} \subseteq \Omega} P(\mathrm{S} \mid i) \min_{j \in \mathrm{S}} V_j(t) \end{cases}$$

- $\mathrm{Q}_i(t)$: backlog of node i at time t
- $P(\mathrm{S}|i)$: probability that **S** be the set of nodes that receive the packet transmitted by node i
- $V_j(t)$: delivery time of node j
- $\min_{j \in S} V_j(t)$: delivery time when **S** reached

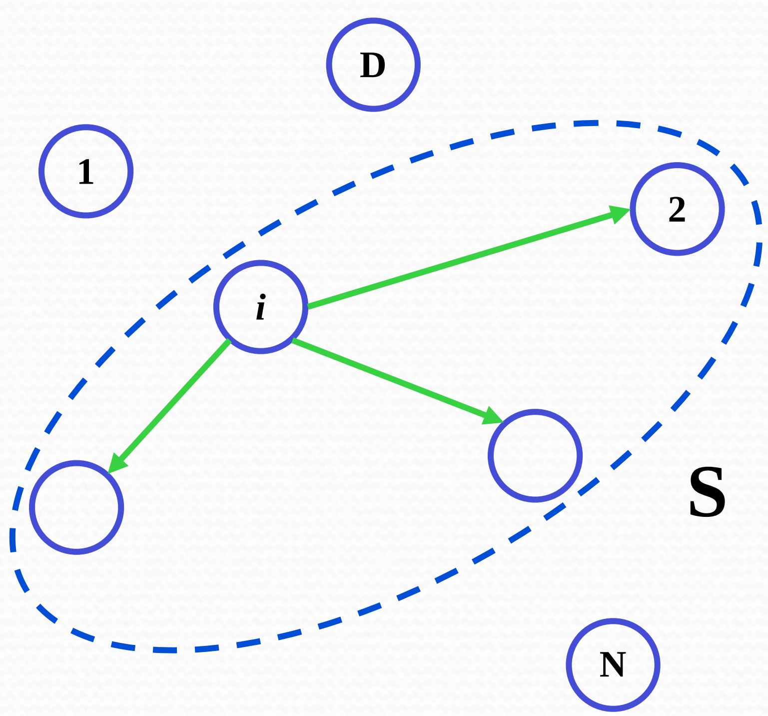


Computation of $V_i(t)$'s

Congestion measures $V_i(t)$ are solutions to the following fixed point equation:

$$\begin{cases} V_D(t) = 0, \\ V_i(t) = \mathrm{Q}_i(t) + \sum_{\mathrm{S} \subseteq \Omega} P(\mathrm{S} \mid i) \min_{j \in \mathrm{S}} V_j(t) \end{cases}$$

- $\mathrm{Q}_i(t)$: backlog of node i at time t
- $P(\mathrm{S}|i)$: probability that **S** be the set of nodes that receive the packet transmitted by node i
- $V_j(t)$: delivery time of node j
- $\min_{j \in S} V_j(t)$: delivery time when **S** reached

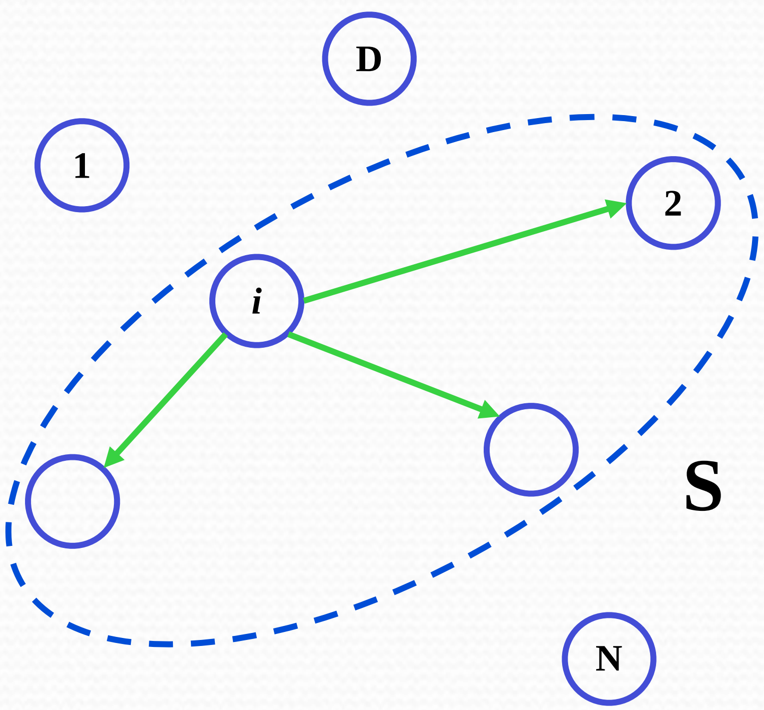


Computation of $V_i(t)$'s

Congestion measures $V_i(t)$ are solutions to the following fixed point equation:

$$\begin{cases} V_D(t) = 0, \\ V_i(t) = \underbrace{Q_i(t)}_{\text{Local congestion at node } i} + \underbrace{\sum_{S \subseteq \Omega} P(S \mid i) \min_{j \in S} V_j(t)}_{\text{average delivery time of the neighbors}} \end{cases}$$

- $Q_i(t)$: backlog of node i at time t
- $P(S|i)$: probability that **S** be the set of nodes that receive the packet transmitted by node i
- $V_j(t)$: delivery time of node j
- $\min_{j \in S} V_j(t)$: delivery time when **S** reached



Computation of $V_i(t)$'s (Cont.)

Computation of $V_i(t)$'s (Cont.)

Centralized Computation:

Computation of $V_i(t)$'s (Cont.)

Centralized Computation:

Stochastic generalization of Dijkstra algorithm

Computation of $V_i(t)$'s (Cont.)

Centralized Computation:

Stochastic generalization of Dijkstra algorithm

1. Initialization:

$$V_{\mathrm{D}}(t) = 0, V_i(t) = \infty \text{ for all } i \in \Omega, i \neq \mathrm{D}$$

$$A = \{\mathrm{D}\},\ A^c \text{ is the complement of } A \text{ with respect to } \Omega.$$

2. Computation:

$$J_i(t) = \frac{1}{P(i,A)}\left[\mathrm{Q}_i(t) + \sum_{S:S\cap A\neq\phi} P(S\,|\,i)\min_{j\in S\cap A} V_j(t)\right], \qquad i \in A^c,$$

where, $P(i,A) = \sum_{S:S\cap A\neq\phi} P(S\,|\,i).$

3. Updtae:

$$i^* = \arg\min_{i\in A^c} J_i(t),\ V_{i*}(t) = J_{i*}(t),\ A = A \cup \{i^*\}.$$

4. Repeat steps 2 and 3 until $A=\Omega$

Computation of $V_i(t)$'s (Cont.)

Centralized Computation:

Stochastic generalization of Dijkstra algorithm

- Centralized controller is responsible for:
  - Collecting backlog information of all nodes in the network
  - Computing congestion measures $V_i(t)$ (worst-case run time $O(N^2)$)
  - Providing all nodes with the results of the computations

Computation of $V_i(t)$'s (Cont.)

Centralized Computation:

Stochastic generalization of Dijkstra algorithm

- Centralized controller is responsible for:
  - Collecting backlog information of all nodes in the network
  - Computing congestion measures $V_i(t)$ (worst-case run time $O(\mathrm{N}^2)$)
  - Providing all nodes with the results of the computations

It is not practical to compute $V_i(t)$ on every time slot.

Computation of $V_i(t)$'s (Cont.)

Computation of $V_i(t)$'s (Cont.)

- ORCD with infrequent computations (**Infreq-ORCD**)
  - Computation of congestion measures V_i is done every T slots
  - Routing decisions at time $nT \le t < (n+1)T$ are based on $V_i(nT)$

Computation of $V_i(t)$'s (Cont.)

- ORCD with infrequent computations (**Infreq-ORCD**)
  - Computation of congestion measures V_i is done every T slots
  - Routing decisions at time $nT \leq t < (n+1)T$ are based on $V_i(nT)$

- For sufficiently large T, the centralized controller has enough time to collect information and flood its decisions to the nodes in the network
- As expected, this lower overhead sacrifices the performance of ORCD

Computation of $V_i(t)$'s (Cont.)

Computation of $V_i(t)$'s (Cont.)

Iterative and Decentralized Computation:

Computation of $V_i(t)$'s (Cont.)

Iterative and Decentralized Computation:

Stochastic generalization of distributed Bellman-Ford algorithm

Computation of $V_i(t)$'s (Cont.)

Iterative and Decentralized Computation:

Stochastic generalization of distributed Bellman-Ford algorithm

$$\widetilde{V}_i^{\,k}(t) = \mathrm{Q}_i(t) + \sum_{S \subseteq \Omega} P(S \mid i) \min_{j \in S} \widetilde{V}_j^{\,k-1}(t), \quad \text{for } k = 1, 2, \ldots$$

Computation of $V_i(t)$'s (Cont.)

Iterative and Decentralized Computation:

Stochastic generalization of distributed Bellman-Ford algorithm

$$\tilde{V}_i^k(t) = Q_i(t) + \sum_{S \subseteq \Omega} P(S \mid i) \min_{j \in S} \tilde{V}_j^{k-1}(t), \quad \text{for } k = 1, 2, \ldots$$

- In each iteration k, all nodes are responsible for:
  - Exchanging $\tilde{V}_i^{k-1}(t)$ with their neighbors
  - Computing $\tilde{V}_i^k(t)$ using equation above

Computation of $V_i(t)$'s (Cont.)

Iterative and Decentralized Computation:

Stochastic generalization of distributed Bellman-Ford algorithm

$$\widetilde{V}_i^{k}(t) = Q_i(t) + \sum_{S \subseteq \Omega} P(S \mid i) \min_{j \in S} \widetilde{V}_j^{k-1}(t), \quad \text{for } k = 1, 2, \ldots$$

- In each iteration k, all nodes are responsible for:
  - Exchanging $\widetilde{V}_i^{k-1}(t)$ with their neighbors
  - Computing $\widetilde{V}_i^{k}(t)$ using equation above

$$\boxed{\lim_{k \to \infty} \widetilde{V}_i^{k}(t) = V_i(t)}$$

Computation of $V_i(t)$'s (Cont.)

Iterative and Decentralized Computation:

Stochastic generalization of distributed Bellman-Ford algorithm

$$\tilde{V}_i^k(t) = Q_i(t) + \sum_{S \subseteq \Omega} P(S \mid i) \min_{j \in S} \tilde{V}_j^{k-1}(t), \quad \text{for } k = 1, 2, \ldots$$

- In each iteration k, all nodes are responsible for:
  - Exchanging $\tilde{V}_i^{k-1}(t)$ with their neighbors
  - Computing $\tilde{V}_i^k(t)$ using equation above

$$\boxed{\lim_{k \rightarrow \infty} \tilde{V}_i^k(t) = V_i(t)}$$

High overhead for each time slot *t*

Computation of $V_i(t)$'s (Cont.)

Iterative and Decentralized Computation:

Stochastic generalization of distributed Bellman-Ford algorithm

$$\tilde{V}_i^k(t) = Q_i(t) + \sum_{S \subseteq \Omega} P(S \mid i) \min_{j \in S} \tilde{V}_j^{k-1}(t), \quad \text{for } k = 1, 2, \ldots$$

- In each iteration k, all nodes are responsible for:
  - Exchanging $\tilde{V}_i^{k-1}(t)$ with their neighbors
  - Computing $\tilde{V}_i^k(t)$ using equation above

$$\boxed{\lim_{k \to \infty} \tilde{V}_i^k(t) = V_i(t)}$$

High overhead for each time slot *t*

It is not practical to compute $V_i(t)$ using this algorithm.

Computation of $V_i(t)$'s (Cont.)

Computation of $V_i(t)$'s (Cont.)

- Iterative ORCD with finite-round computations
  - Number of iterations is limited to some K rounds :

$$\tilde{V}_i^k(t) = \mathrm{Q}_i(t) + \sum_{S \subseteq \Omega} P(S \mid i) \min_{j \in S} \tilde{V}_j^{k-1}(t), \quad \text{for} \quad 1 \le k \le \mathrm{K},$$

$$\tilde{V}_i^0(t) = \tilde{V}_i^{\mathrm{K}}(t-1).$$

Computation of $V_i(t)$'s (Cont.)

- Iterative ORCD with finite-round computations
  - Number of iterations is limited to some K rounds :

$$\widetilde{V}_i^k(t) = \mathrm{Q}_i(t) + \sum_{S \subseteq \Omega} P(S \mid i) \min_{j \in S} \widetilde{V}_j^{k-1}(t), \quad \text{for} \quad 1 \le k \le \mathrm{K},$$

$$\widetilde{V}_i^0(t) = \widetilde{V}_i^{\mathrm{K}}(t-1).$$

- Distributed ORCD (**D-ORCD**)
  - Special case when K=1:

$$\widetilde{V}_i(t) = \mathrm{Q}_i(t) + \sum_{S \subseteq \Omega} P(S \mid i) \min_{j \in S} \widetilde{V}_j(t-1).$$

Simulations

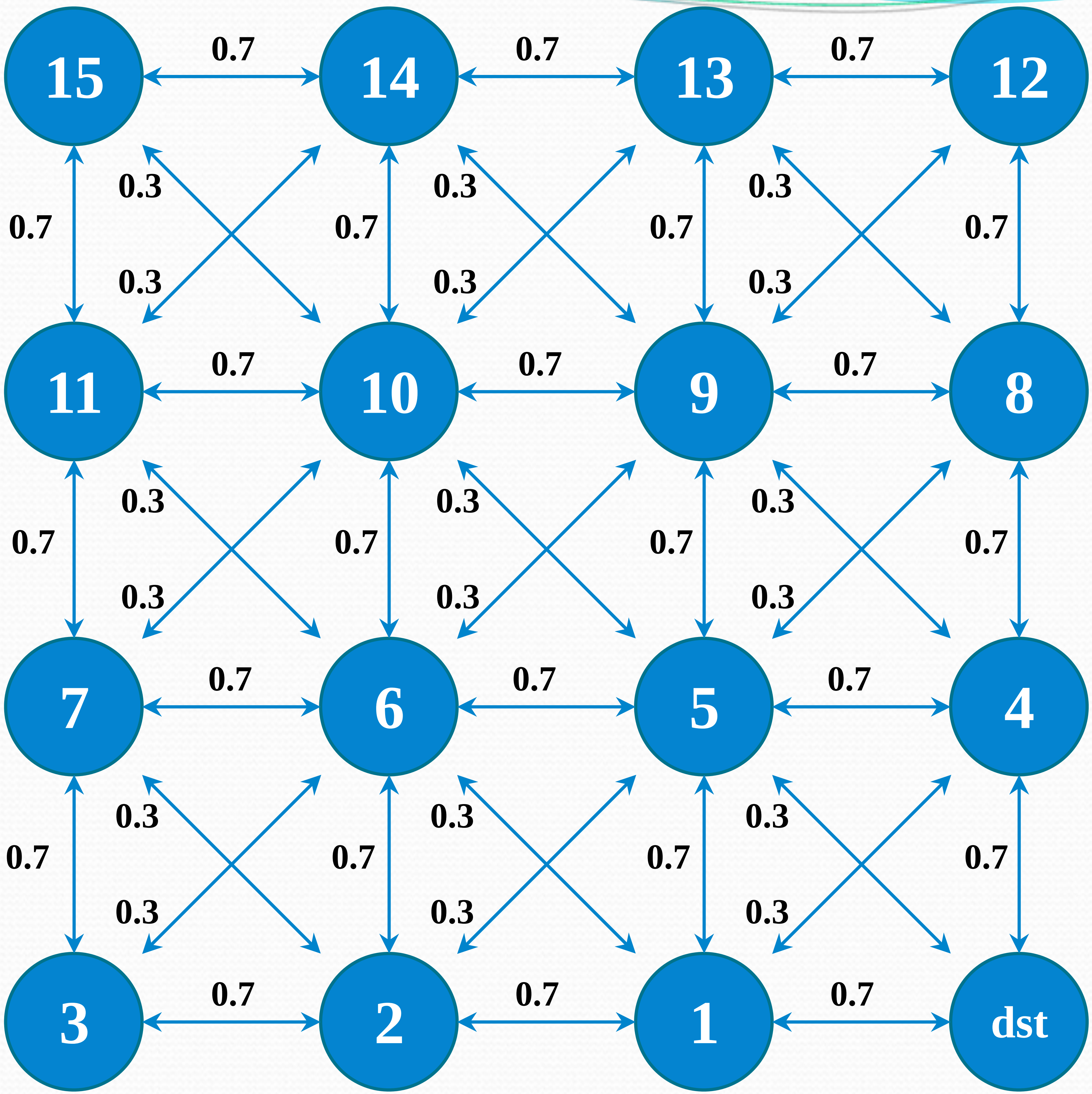
Simulations

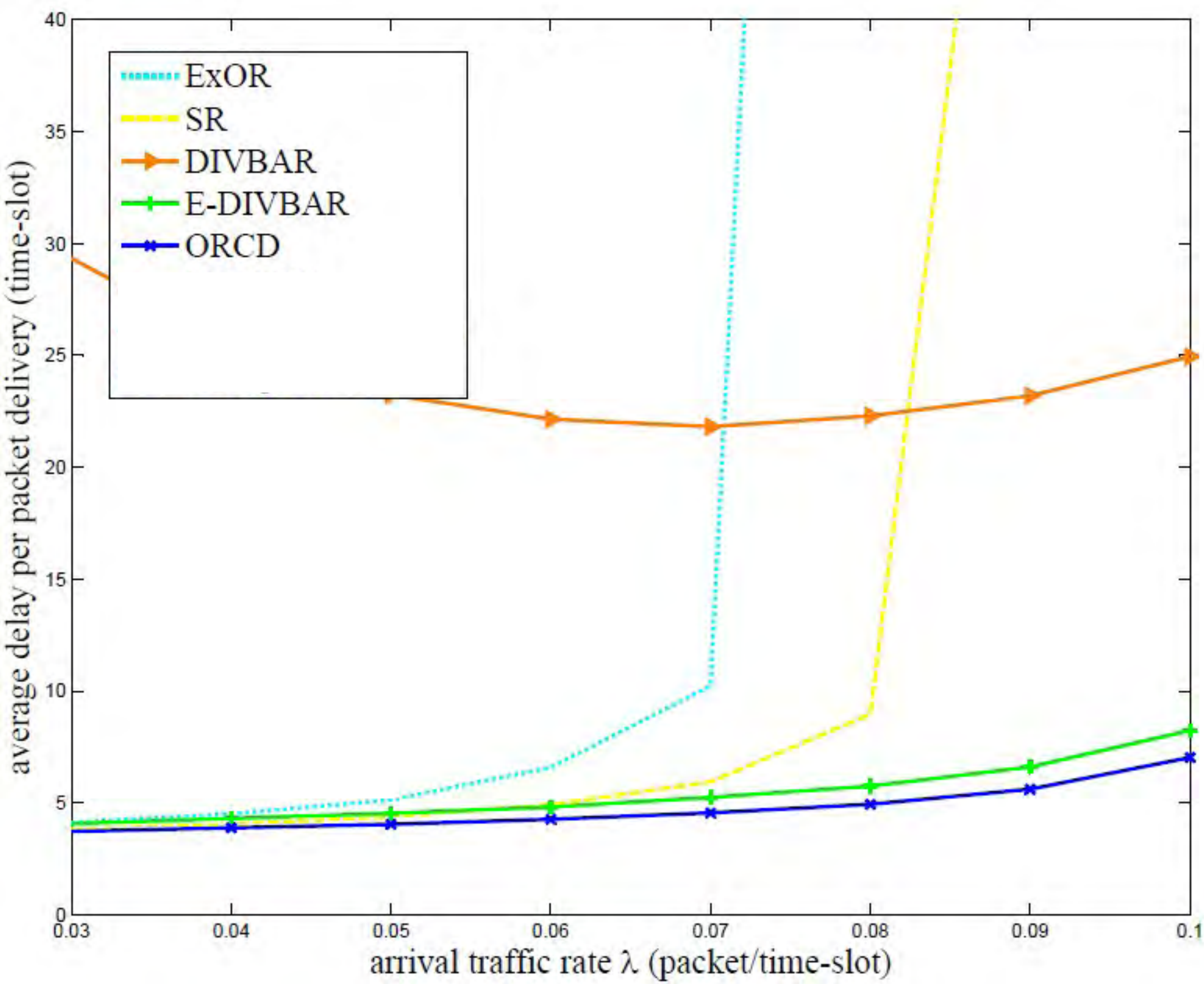
- Comparing the delay performance of:
  - **ExOR**, **SR** (expected hop-counts to the destination)
  - **DIVBAR** (queue backlog)
  - **E-DIVBAR** (queue backlog + expected hop-counts)
  - **ORCD**, **Infreq-ORCD**, and **D-ORCD** (expected delivery time)

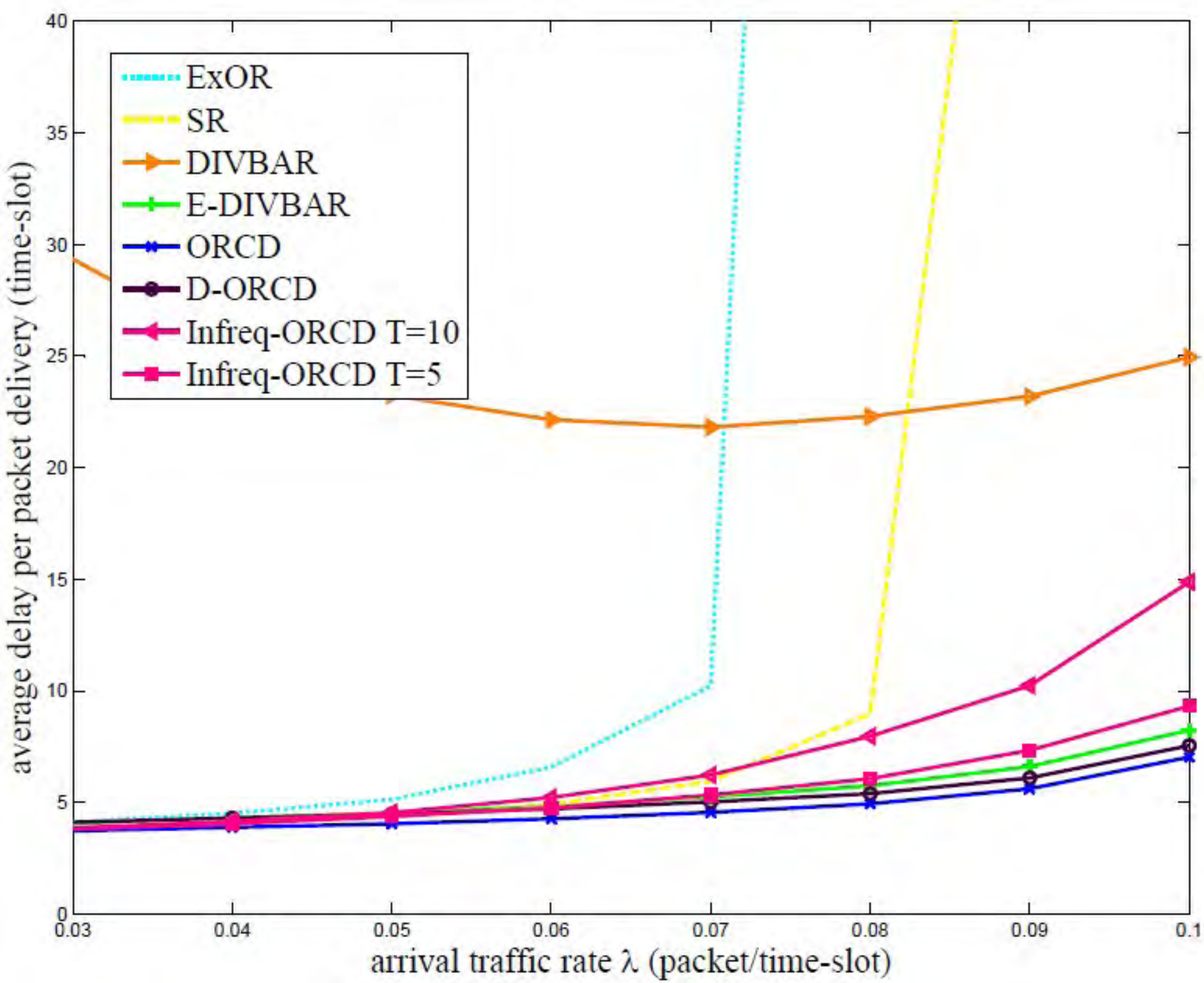
Simulations

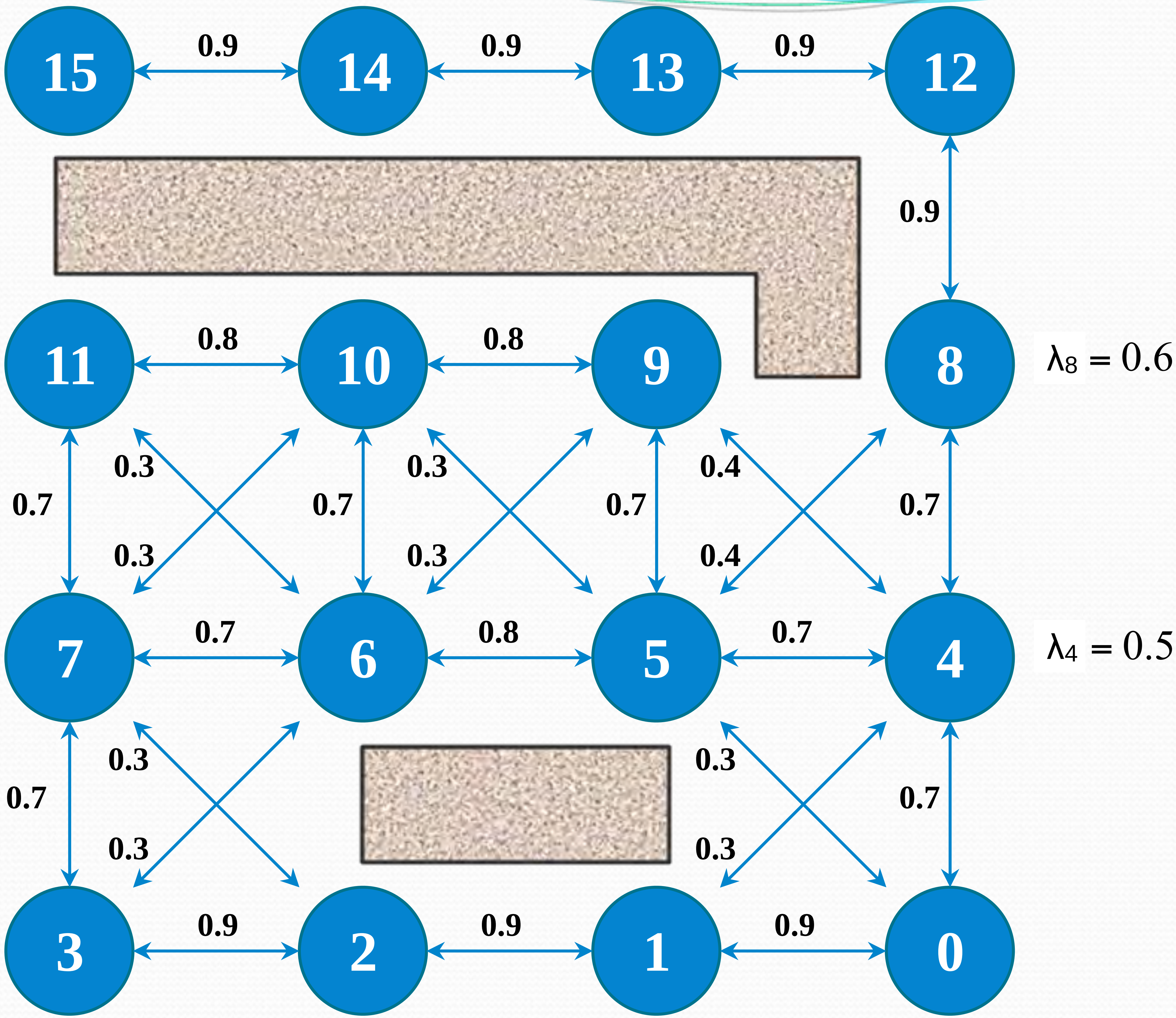
- Comparing the delay performance of:
  - **ExOR**, **SR** (expected hop-counts to the destination)
  - **DIVBAR** (queue backlog)
  - **E-DIVBAR** (queue backlog + expected hop-counts)
  - **ORCD**, **Infreq-ORCD**, and **D-ORCD** (expected delivery time)

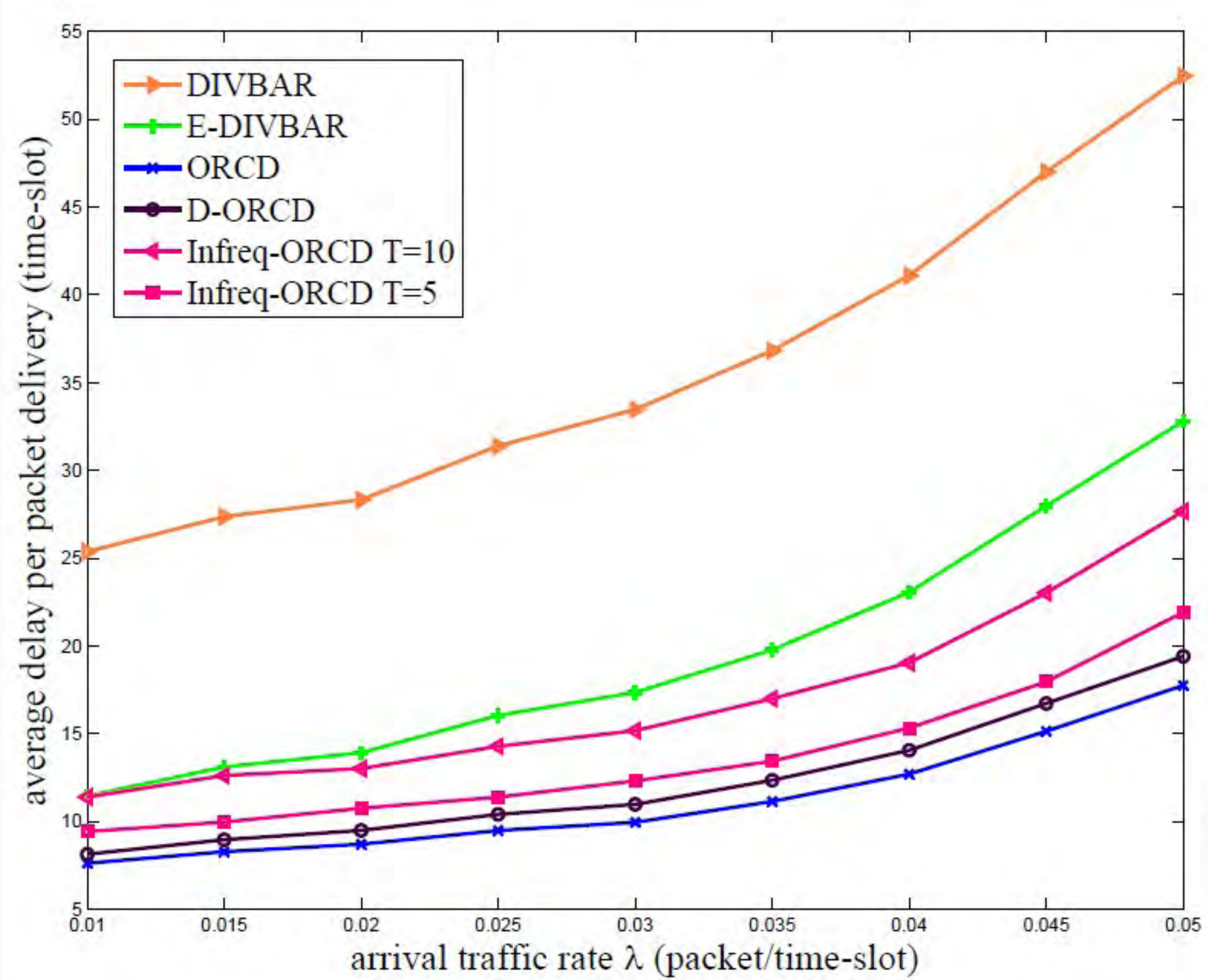
Routing Policy	Throughput Optimal	Delay Performance
ExOR, SR	✘	Poor delay in high traffic
DIVBAR	✓	Poor delay in low traffic
E-DIVBAR	✓	?
ORCD & its variants	✓	?


15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
dst
0.7
0.3


ExOR
SR
DIVBAR
E-DIVBAR
ORCD
average delay per packet delivery (time-slot)
arrival traffic rate λ (packet/time-slot)
0
5
10
15
20
25
30
35
40
0.03
0.04
0.05
0.06
0.07
0.08
0.09
0.1


ExOR
SR
DIVBAR
E-DIVBAR
ORCD
D-ORCD
Infreq-ORCD T=10
Infreq-ORCD T=5
average delay per packet delivery (time-slot)
arrival traffic rate λ (packet/time-slot)
0
5
10
15
20
25
30
35
40
0.03
0.04
0.05
0.06
0.07
0.08
0.09
0.1


15
14
13
12
0.9
0.9
0.9
0.9
11
10
9
8
0.8
0.8
$\lambda_8 = 0.6$
0.7
0.3
0.3
0.7
0.3
0.3
0.7
0.4
0.4
0.7
7
6
5
4
0.7
0.8
0.7
$\lambda_4 = 0.5$
0.7
0.3
0.3
0.3
0.3
0.7
3
2
1
0
0.9
0.9
0.9



DIVBAR
E-DIVBAR
ORCD
D-ORCD
Infreq-ORCD T=10
Infreq-ORCD T=5
average delay per packet delivery (time-slot)
arrival traffic rate λ (packet/time-slot)
55
50
45
40
35
30
25
20
15
10
5
0.01
0.015
0.02
0.025
0.03
0.035
0.04
0.045
0.05

Summary

Summary

Routing Policy	Throughput Optimal	Delay Performance
ExOR, SR	✘	Poor delay in high traffic
DIVBAR	✓	Poor delay in low traffic
E-DIVBAR	✓	Depends on topology & traffic
ORCD	✓	Good delay performance in all traffic conditions
Infreq-ORCD	✓	Depends on network & traffic
D-ORCD	?	Good delay performance in all traffic conditions

Future and On-going Research

- Extensions:
  - multi-rate and multi-commodity
  - Ack explosion: limiting neighbor set
- Interference: scheduled MAC vs. random access
- Multi-user detection, cooperation, fancy PHY
- Network coding