Finding an Optimal Routing Policy on Simple Client-Server Models

Stephane Durocher
Department of Computer Science
University of British Columbia
2366 Main Mall
Vancouver, B.C., Canada V6T 1Z4
durocher@cs.ubc.ca

Nicholas Pippenger
Department of Computer Science
University of British Columbia
nicholas@cs.ubc.ca

January 26, 1999

Abstract

We examine the problem of finding an optimal policy in routing single-connection calls between a client and a server on simple models of connection networks. We study the graphs $K_{2,2}$, K_5 and $K_{3,3}$ with respect to possible configurations of vertices, edges and directed edges within our client-server model. The many different configurations of the system are reduced to a small set of equivalence classes of possible states. The transitions between states are examined with respect to variable call request rates and call hang-up rates, λ and γ, respectively. We develop a technique to measure the optimality of a policy by counting the expected number of blocked calls from a given state in λ and γ at degrees 0, 1 and 2. We show that the optimality of a routing policy sometimes depends upon traffic; there exist decisions which are optimal within a specific range of call request and hang-up rates with the opposite decision being optimal outside of that range.

1 Problem Definition

1.1 Problem and Motivation

Network clients need to communicate. Whether the medium be telephone or internet, the first step in communication requires connection to a local server which, in turn, connects to some higher-level network. Thus, a client requests to make a call and, if all goes well, a neighbouring server routes the call. Often, more than one server is available; in such a case, a routing decision must be made as to which server should handle the call. The outcome of this decision affects the state of the local network. How should such a decision be made? How does one determine an optimal policy for a given network?

*This work was supported by an NSERC Research Grant.

1.2 Previous Work

In previous work, Beneš studied the importance of reducing states using symmetry within the network graph. He found optimal routing policies for several small telephone switch layouts including the tetrahedral network graph, K_4. Beneš conjectured that "optimal policy is basically a combinatorial feature of the network alone."[Ben66] Contrary to this, we show that optimality sometimes depends upon traffic; there exist decisions that are optimal within a specific range of call request and hang-up rates, λ and γ, with the opposite decision being optimal outside of that range.

1.3 Definition of Terms

A *client* is a node in the network graph whose only neighbours are servers. All clients are identical and request service from a server at a common rate, λ. Clients terminate calls at a common rate, γ. A client may only be engaged in a single call at any time.
A *server* is a node in the network graph whose only neighbours are clients. All servers are identical, hence, any server neighbouring a client may serve a call request from that client. A server may only be connected to a single client at any time. Any requests made to a server while it is busy are blocked.
A server or client is *busy* when it is currently engaged in a call.
A server or client is *idle* or *free* whenever it is not busy.
A *call request* is placed by a client to a server when requesting service for a call.
A *hang-up* is the termination of a call; it is initiated by the client. Both client and server are again free to engage in new connections.

A *call* may take place between any neighbouring pair consisting of a server and a client so long as both the client and server are free of other current calls. A call is the result of a call request from a client to a free server which accepts the request.
A *blocked call* results when a client places a call request to a busy server. The call is refused; no connection is established.
An *event* is either a call request or a hang-up.
A routing *decision* must be made whenever a client requesting service has two free neighbouring servers. One of the two servers may serve the request. The outcome of this decision affects the availability of future service to other clients.
A *policy* determines how decisions are to be made in assigning servers to call requests.
An *optimal* policy is achieved by minimizing the expected number of blocked calls or maximizing the expected number of successful calls. Since a server can only connect a single call, equivalently, we seek to maximize the expected number of busy servers.
The *state* of the network describes a particular configuration of busy servers and clients embedded within the graph.

1.4 Modelling the Client-Server Network

We limit our study to simple regular graphs. This allows most configurations of the graph to be reduced to a small number of equivalence classes for the different possible states. Furthermore, we only consider networks in which clients have exactly two servers as neighbours; this permits a more compact graph model in which each vertex represents a server and each edge represents a client.

As an example, figure 1 shows two representations of the tetrahedral network. This network consists of the complete graph K_4, which is composed of 4 servers and 6 clients. In the network graph on the left, clients appear as white vertices and servers as black. In the next graph, the client vertices are removed and clients are simply represented by edges. We use this second representation.

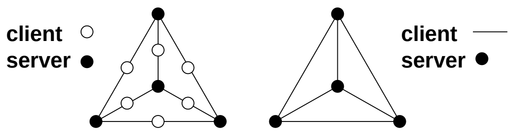

Figure 1. tetrahedral client-server network

During a call, a client receives service from one of the two servers at the ends of the edge which represents it. We represent a client and server as being engaged in a call by directing the client's edge toward the server being used. See figure 2.

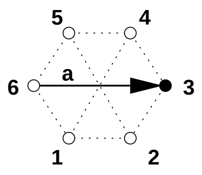

Figure 2. client a is connected to server 3.

2 Examining the Client-Server Model

2.1 Choice of Graphs

Even simple network graphs produce large numbers of possible configurations. In order to study the states of the system, one must rely on symmetry within the graph to bring the number of states to within a manageable range. Thus we choose regular graphs with a high degree of symmetry.

We gain insight into the complexity of the graph by counting the numbers of busy server states, busy client states and busy client-server pair states. The two former help determine the suitability of a graph for analysis whereas the latter will be used in our study of routing policy decisions.

Since vertices correspond to servers, we count the number of busy server states by counting the number of vertex combinations possible. Similarly, since edges correspond to clients, we count the number of busy client states by enumerating the different embeddings of connected components into the graph. See figure 24 in the appendix for the busy client states and busy server states for $K_{3,3}$.

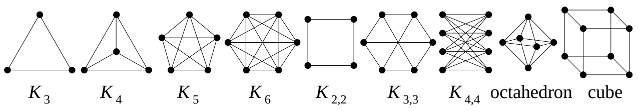

Figure 3. K_3, K_4, K_5, K_6, $K_{2,2}$, $K_{3,3}$, $K_{4,4}$, octahedron and cube

Graph State Properties

graph	$\|V\|$	$\|E\|$	d	busy server states	busy client states	client-server pairs	transition decisions
K_3	3	3	2	4	4	5	1
$K_{2,2}$	4	4	2	6	6	9	3
K_4	4	6	3	5	11	12	3
K_5	5	10	4	6	39	29	10
$K_{3,3}$	6	9	3	10	26	45	27
octa	6	12	4	10	>100		
K_6	6	12	5	7	>100		
cube	8	12	3	22	>100		
$K_{4,4}$	8	16	4	15	>100		

The above table displays $|V|$, $|E|$ and d, the number of vertices, edges and the degree of each graph along with the order of busy server, busy client and client-server pair states. Also listed are the number of transition decisions for each graph.

As graphs grow the number of possible client-server pairs grows proportionally to the number of busy client states. As this occurs, the number of decisions also grows rapidly. K_3, $K_{2,2}$ and K_4 each have few client-server pairs and few decisions making examination quite possible but uninteresting. $K_{3,3}$ and K_5 jump to 10 and 27 decisions, respectively, while keeping the number of client-server states within manageability. Moving a step higher to K_6, $K_{4,4}$, the cube or the octahedron increases the number of client states to greater than 100 (the authors stopped counting at that point). With such a large number of busy client states, the number of client-server states becomes immense, rendering the study of state transitions near to impossible.

$K_{3,3}$ and K_5, therefore, both provide an interesting assortment of decisions to examine while remaining within a manageable number of states. The major differences between the two lie in the complete bipartite properties of $K_{3,3}$. $K_{3,3}$ is a subgraph of K_6. Having fewer edges and more vertices than K_5 makes the state space for $K_{3,3}$ quite interesting.

We attempt to find optimal policies for $K_{2,2}$, K_5 and $K_{3,3}$. The techniques used to solve for K_5 and $K_{3,3}$ are identical. Thus, we only discuss $K_{2,2}$ and $K_{3,3}$ in detail; we present results for all three networks in the appendices.

2.2 Group Symmetries

Although it seems to be a simple graph, $K_{3,3}$ produces a very large number of possible configurations of the system. We quickly note that many of these are equivalent. For example, figure 4 shows two scenarios in which a single client-server pair are involved in a call. These two scenarios are different, yet we would like to consider them to be the same. We note that the second graph is a single left rotation of the first by $\pi/3$. We would like to find all symmetries of the graph such that vertex adjacency is maintained.

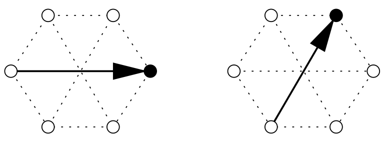
Figure 4. equivalent states of the system

Group theory quickly comes into play and we find ourselves with a group of order 72, which is a subgroup of the symmetric group, S_6. Two operations form a basis for the

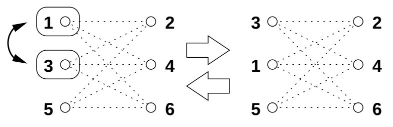

Figure 5. exchanging non-adj. vertices: x

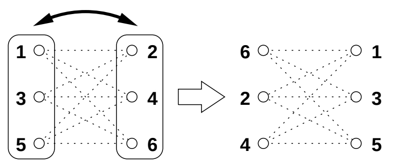

Figure 6. rotating partitions: r

group; these are pictured in figures 5 and 6. The exchange operation, x, interchanges two non-adjacent vertices in the graph. Any two vertices within the same partition may be switched without affecting adjacency. The rotation operation, r, interchanges the two entire partitions. Again, since the graph is complete bipartite, adjacency is unaffected. Figure 7 displays the hexagonal representation of the operations. Note that $x^2 = e$ and $r^6 = e$ where e is the identity transformation.

Each operation is isomorphic to an element of S_6. x maps to (13) and r maps to (123456). Any permutation which maintains vertex adjacency in $K_{3,3}$ can be derived from some composition of x and r. Some of these transformations are pictured in figure 8. Together, $\{x, r\}$ generate the group of all symmetries within $K_{3,3}$ such that all vertices which are initially adjacent remain adjacent after the transformation.

Since we know that there are $3!$ permutations of the left partition, $3!$ permutations of the right partion and one interchange of partitions possible, we derive that the group has order $6 \times 6 \times 2 = 72$.

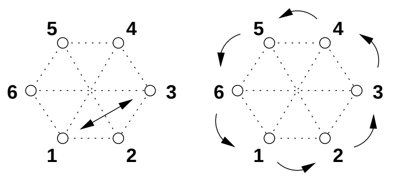

Figure 7. hexagonal representation of x and r

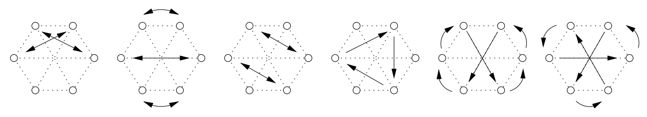

Figure 8. other permutations in the group

In general, the permutation group for K_n is S_n. We know $|S_n| = n!$. The permutation group for $K_{n,n}$ is a subgroup of S_n with $|S_{K_{n,n}}| = 2(n!)^2$.

We map configurations of the system into equivalence classes of states. Any two network configurations which are equivalent under some transformation within the group form a single state. For example, all three configurations in figure 9 belong to the same equivalence class. These all represent the same state of the system.

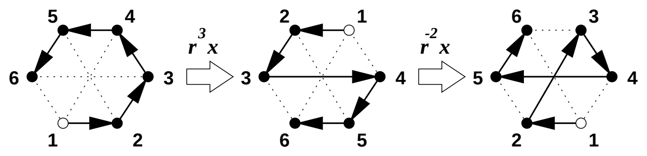

Figure 9. r^3x followed by $r^{-2}x$

In total, we derive 45 possible states. These are listed in figure 25 in the appendix. The system moves from one state to another whenever a server accepts a call request from a client or whenever a client hangs-up an existing call to a server. The system remains in the same state whenever a call request is blocked. Any event other than a blocked call results in a change of state in the system.

Under a given policy, the states and their allowed transitions form a finite state machine. Crossing $E \times C$ forms the corresponding input language, where E are the allowed events, namely, call request or hang-up, and C is the set of clients, $\{a, \ldots, i\}$.

2.3 State Transitions

Events within the system occur at variable rates. We define λ to be the rate at which a client makes a call request. We define γ to be the rate at which a busy client terminates its current call. Each of these rates is measured per some fixed unit time.

The current state of the system determines possible transitions upward or downward in the state hierarchy. In a given state there are c calls in progress, $0 \leq c \leq 6$. Any one of these may terminate meaning that there are c possible ways of moving down to a lower state with $c-1$ connections. Similarly, there are $9-c-b$ possible ways of moving up to a higher state with $c+1$ connections, where b is the number of blocked calls, $0 \leq b \leq 9-c$. Often, more than one of these paths lead to a common state. All downward transition rates from a state must sum to $c\gamma$ and all upward transition rates must sum to $(9-c-b)\lambda$.

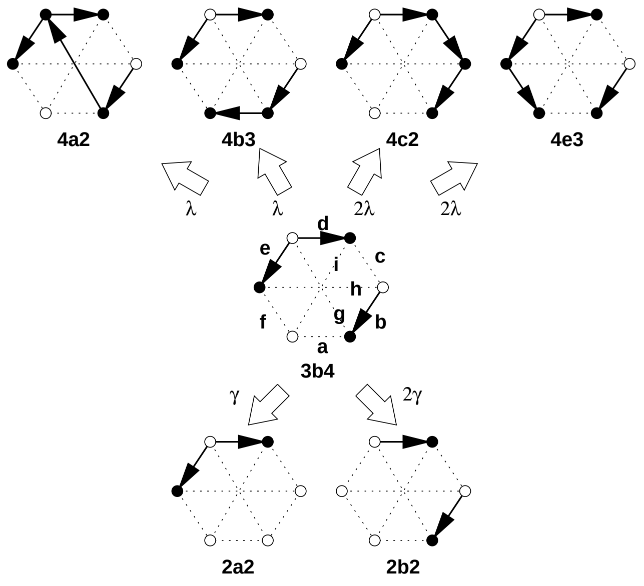

Figure 10. all transitions out from state $3b4$

For example, figure 10 displays all forward and backward transitions out of state $3b4$. Since there are three calls in progress, there are three ways of moving down in the state hierarchy. Clients b, d and e are involved in calls. A hang-up from client b moves the system to state $2a2$; this transition occurs at rate γ. A hang-up from clients d or e moves the system to state $2b2$. Figure 11 shows the two permutations of $2b2$ to which one may arrive from $3b4$. Note that the operation $(46) = r^3xr^3$ transforms one permutation into the other. Since there are two ways of going from states $3b4$ to $2b2$, this transition occurs at rate 2γ.

There are $9-3-0=6$ free clients which may place

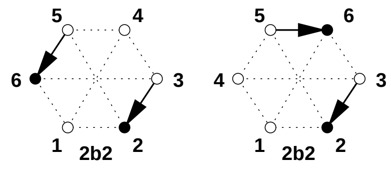

Figure 11. two permutations of $2b2$

calls requests. None of the possible calls out from $3b4$ are blocked. Thus, any free edge which places a call request has an idle server at one end which may take the call. Again, some of the calls from different clients result in the same state transition. A call request from client g moves the system to state $4a2$ and occurs at a rate of λ. A call request from client a moves the system to state $4b3$ and occurs at a rate of λ. Call requests from clients c or h move the system to state $4c2$ and occur at a rate of 2λ. Call requests from clients f or i move the system to state $4e3$ and occur at a rate of 2λ.

2.4 Policy Decisions

In some cases, a client makes a call request and both of its neighbouring servers are free. When this occurs, we must decide which server should take the call. Since such a routing decision affects the placement of available servers remaining, it has a global effect on the optimality of the system. We denote a decision by d_x, a boolean which takes on the value 0 or 1. $\bar{d}_x$ is the negation of the decision, $\bar{d}_x = 1 - d_x$.

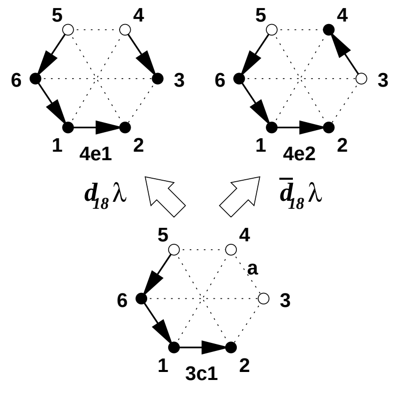

Figure 12. d_{18}: $3c1 \to 4e1$ or $4e2$

Figure 12 shows one of the 27 decisions in the state transitions. In this case, edge a makes a call request with both servers 3 and 4 being free. Choosing server 3 moves the system to state $4e1$, while choosing server 4 moves to state $4e2$.

The set of all decisions makes up the routing policy for the network. Our problem is to find an optimal policy such that we minimize the expected number of blocked calls for given expected rates λ and γ.

Since the routing of a new call requires the decision-making, all decisions occur on forward transitions. Some decisions lead to the same equivalence class regardless of which path is chosen. We assume that the policy arbitrarily picks one of the two symmetric outcomes; these are not actual policy decisions and are not included in the 27 decisions which we study. There are 5 such random symmetric decisions in the state hierarchy. Figure 13 displays one of these.

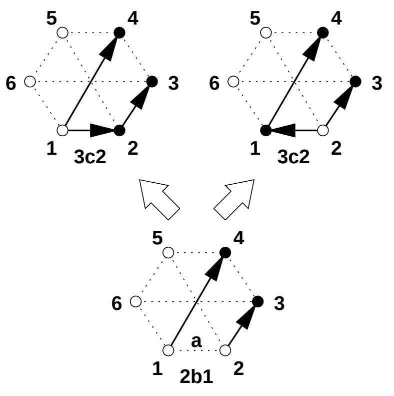

Figure 13. random symmetric decision

Furthermore, the optimality of some policy decisions is deemed obvious by the layout of the busy clients. Such an obvious decision occurs whenever all neighbouring clients of one of the two servers involved are all currently connected to another server. The outcome of the decision renders one of the servers useless. That is, it keeps the server idle without there being any neighbouring clients which may place call requests. Since clients outnumber servers, the advantageous decision always chooses a state in which no server is wasted. 6 of the 27 decisions are obvious decisions. Figure 14 displays one of these; edge a may be served by servers 1 or 2. Choosing server 1 would cause server 2 to be of no use to the system. Thus the obvious decision is to choose server 2.

See the appendix for a table listing all forward and backward transition rates including decision variables.

2.5 Steady State Expectation

For any state, we know the rates of traffic to and from every neighbouring state in the hierarchy. In order to measure the performance of a policy, we must find the expected probability of being in a state in terms of λ and γ. Given this expectation, one may easily calculate the expected number of

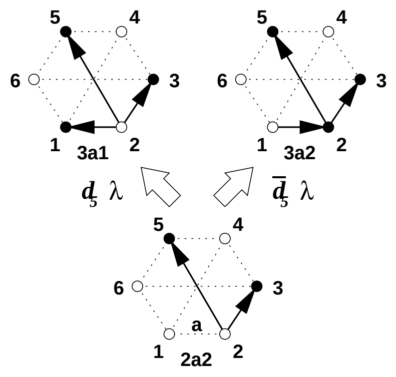

Figure 14. obvious decision

busy servers and thus compare policies for optimality.

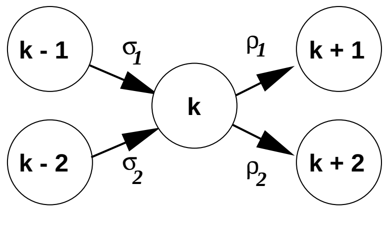

Figure 15. state transitions to and from state k

We define P_k to be the probability that the system is in state k, $0 \leq P_k \leq 1$. Figure 15 shows a state k which receives incoming traffic from states $k-1$ and $k-2$ at rates σ_1 and σ_2, respectively, and sends outgoing traffic to states $k+1$ and $k+2$ at rates ρ_1 and ρ_2, respectively. We know that traffic into P_k must be equal to traffic out from P_k. We write down the following traffic flow equilibrium equation:

$$P_{k-1}\sigma_1 + P_{k-2}\sigma_2 = P_k(\rho_1 + \rho_2)$$

$$\Leftrightarrow P_{k-1}\sigma_1 + P_{k-2}\sigma_2 + P_k(-\rho_1 - \rho_2) = 0 \qquad (1)$$

Note that this situation is over-simplified since in our case, the existence of a transition from one state to the next usually implies that the reverse transition back into the state of origin must also exist; this is always true except for decision transitions which are never followed.

For m states we write m such distinct linear equations in terms of P_1 through P_m and the rates of traffic between them. These m equations are linearly dependent. To eliminate linear dependence and solve for $\{P_1 \ldots, P_m\}$, we use the fact that since $\{P_1 \ldots, P_m\}$ are probabilities over a common event then,

$$\sum_{i=1}^{m} P_i = 1 \qquad (2)$$

3 Method of Analysis

3.1 Computational versus Theoretic Methods

Routing policies for small graphs contain few decisions. The routing policies for $K_{2,2}$ and K_4 each have three routing decisions meaning that there are $2^3 = 8$ possible policies. For each of these eight policies we can calculate and compare the expected number of busy servers over varying λ and γ. For $K_{3,3}$, however, we have 27 routing decisions and, therefore, $2^{27} = 134,217,728$ possible policies. Furthermore, as we will see, no one single policy remains maximal over all λ and γ. Such exponential growth means that finding the optimal policy for larger networks requires theoretical analysis as opposed to simple direct computation.

3.2 Using a Computational Method to Find the Optimal Policy for $K_{2,2}$

To illustrate the use of a computational method, we find the optimal policy for $K_{2,2}$. Figure 16 displays the allowed state transitions along with decision variables and transition rates for $K_{2,2}$. From equation 1 we get the following:

$$P_2\gamma = P_1 4\lambda$$

$$P_1 4\lambda + P_3 2\gamma + P_4 2\gamma + P_5 2\gamma + P_6 2\gamma =$$
$$P_2(\gamma + d_1\lambda + (1+d_2)\lambda + \bar{d}_1\lambda + \bar{d}_2\lambda)$$
$$P_2 d_1\lambda + P_7\gamma = P_3(2\gamma + 2\lambda)$$
$$P_2(1+d_2)\lambda + P_7 2\gamma + P_8\gamma = P_4(2\gamma + (1+d_3)\lambda + \bar{d}_3\lambda)$$
$$P_2\bar{d}_1\lambda + P_8\gamma = P_5(2\gamma + \lambda)$$
$$P_2\bar{d}_2\lambda + P_8\gamma = P_6(2\gamma + 2\lambda)$$
$$P_3 2\lambda + P_4(1+d_3)\lambda + P_9 4\gamma = P_7(\gamma + 2\gamma + \lambda)$$
$$P_4\bar{d}_3\lambda + P_5\lambda + P_6 2\lambda = P_8(\gamma + \gamma + \gamma)$$
$$P_7\lambda = P_9 4\gamma$$

To remove linear dependence, we replace the last equation using equation 2. We write the equations in matrix form. See figure 17 for the matrix.

The right sides of the equations are,

$$z = [\ 0, 0, 0, 0, 0, 0, 0, 0, 1\ ]$$

We wish to solve for,

$$a = [\ P_1, P_2, P_3, P_4, P_5, P_6, P_7, P_8, P_9\ ]$$

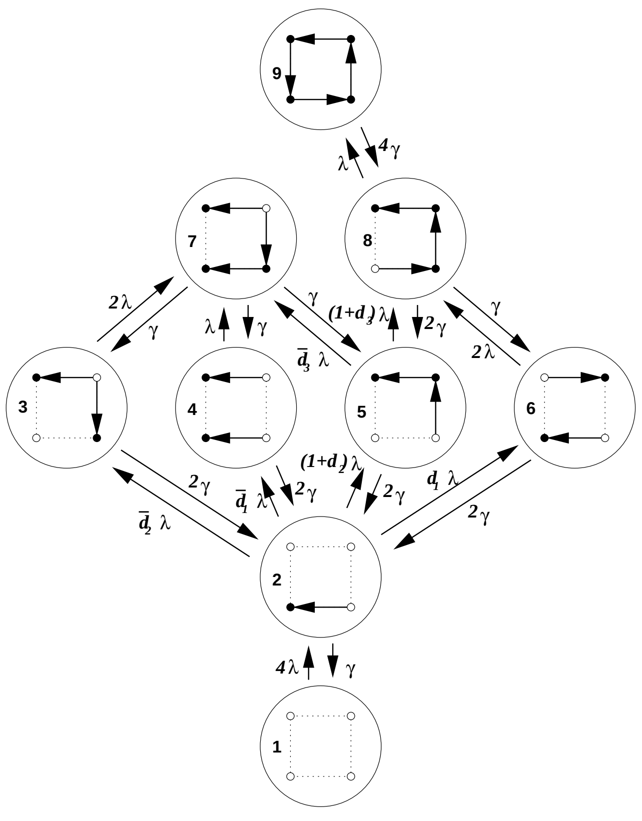

Figure 16. state transitions for $K_{2,2}$

where,

$$Qa^T = z^T \tag{3}$$

a gives us the expectations $\{P_1, \ldots, P_m\}$. We then multiply these probabilities by the number of busy servers in each state, s,

$$s = [\ 0\ ,\ 1\ ,\ 2\ ,\ 2\ ,\ 2\ ,\ 2\ ,\ 3\ ,\ 3\ ,\ 4\ ]$$

$$v_d = a^T s \tag{4}$$

Given the decisions $d = \{d_1, d_2, d_3\}$, we can thus solve for the expected number of busy servers for variable λ and γ for any policy on $K_{2,2}$.

In this way, we compare all eight policies. Policy $d_1 = 1, d_2 = 1, d_3 = 1$ has its number of expected busy servers given by,

$$v_{1,1,1} = 4\frac{\lambda}{\lambda + 1}$$

Solving for all eight expressions, v_{d_1,d_2,d_3}, and comparing the results, we find that,

$$\forall \lambda > 0\ \forall d \in \{0,1\}^3\ v_d \leq v_{1,1,1} \tag{5}$$

$$Q = \begin{bmatrix} -4\lambda & \gamma & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 4\lambda & -\gamma - 3\lambda & 2\gamma & 2\gamma & 2\gamma & 2\gamma & 0 & 0 & 0 \\ 0 & d_1\lambda & -2\gamma - 2\lambda & 0 & 0 & 0 & \gamma & 0 & 0 \\ 0 & (1+d_2)\lambda & 0 & -2\gamma - 2\lambda & 0 & 0 & 2\gamma & \gamma & 0 \\ 0 & \bar{d}_1\lambda & 0 & 0 & -2\gamma - \lambda & 0 & 0 & \gamma & 0 \\ 0 & \bar{d}_2\lambda & 0 & 0 & 0 & -2\gamma - 2\lambda & 0 & \gamma & 0 \\ 0 & 0 & 2\lambda & (1+d_3)\lambda & 0 & 0 & -3\gamma - \lambda & 0 & 4\gamma \\ 0 & 0 & 0 & \bar{d}_3\lambda & \lambda & 2\lambda & 0 & -3\gamma & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Figure 17. matrix form of equilibrium equations

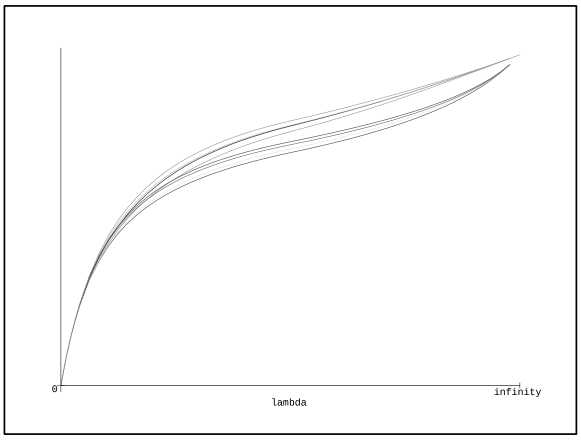

Figure 18. expected number of busy servers for $K_{2,2}$ for all possible policies

That is, for any rate of traffic, we maximize the expected number of busy servers by choosing policy $d_1 = 1, d_2 = 1, d_3 = 1$. Figure 18 displays the functions for all eight policies over varying λ. Note that we eliminate γ by fixing our time unit such that $\gamma = 1$.

Whenever we have a small number of decisions, such a computational method allows us to find a globally optimal policy if one exists. For more complex graphs, such as $K_{3,3}$, however, we must use an analytical approach.

3.3 Examining Blocked Calls for High and Low Traffic in $K_{3,3}$

We study the policy decisions for $K_{3,3}$ by local examination of the states to which each decision leads. We compare these two states by counting the number of blocked calls which may occur in each state. We say that state u has a probability of blocking B_u, $0 \leq B_u \leq 1$. Thus, when deciding between states u and v, we choose the state with a

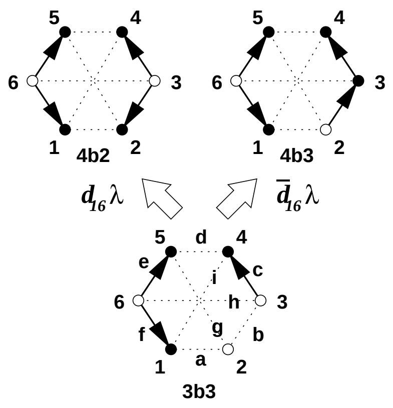

Figure 19. comparing blocking probability in d_{16}

lower probability of blocking, namely, $\min_x B_x$.

Figure 19 displays a routing decision for client b. In this example, either servers 2 or 3 may serve the call. In state $4b2$, four clients are blocked: a, d, i and g. In state $4b3$, however, only two clients are blocked: d and i. When deciding between these two states, therefore, one would prefer the option with lower probability of blocking, namely, $4b3$.

Many such decision have two immediately neighbouring states with differing numbers of possible blocked calls. For $K_{3,3}$, these are $\{d_2, d_4, \ldots, d_{10}, d_{13}, \ldots, d_{16}, d_{18}, \ldots, d_{25}, d_{27}\}$. See figure 27 in the appendix for the decision results.

Included in these are the six obvious decisions, some of which do not actually have differing numbers of possible blocked calls: $\{d_5, d_{15}, d_{19}, d_{23}, d_{25}, d_{27}\}$.

We solve 21 of the 27 decisions in this way. In the six remaining decisions, both resulting states have equal probability of blocking. In order to compare these, we examine not only immediately neighbouring states, but all states which may be reached through a one event jump. That is, for every state, we wish to find the probability of blocking given that one hang-up or one call request occurs before the second call request. We wish to find the probability of blocking over a larger subgraph of the state hierarchy, namely, all states that are distance two or less from the given state.

We know the constant factor B_u for any one state, u. If k is the number of busy servers in a state, then the probability that the next event is a call request is given by,

$$\frac{\lambda(9-k)}{\gamma k + \lambda(9-k)} \tag{6}$$

The probability that the next event is a hang-up is given by,

$$\frac{\gamma k}{\gamma k + \lambda(9-k)} \tag{7}$$

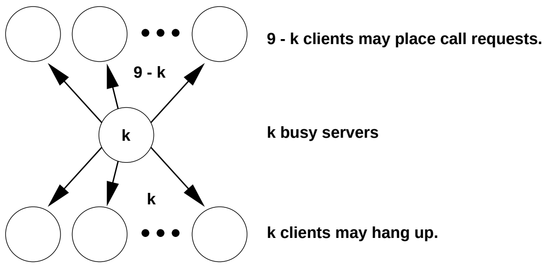

Figure 20. moving up or down in the hierarchy

Note that this includes the probability that the next event is a call request that is blocked. We can find a weighted average of the probabilities of blocking for the states directly above and below any state and combine them using (6) and (7).

For example, consider the state transitions displayed in figure 10. State $3b4$ has three busy servers. Thus, the probability that the next event is a call request is,

$$\frac{\lambda(9-3)}{3\gamma + \lambda(9-3)} = \frac{2\lambda}{\gamma + 2\lambda} \tag{8}$$

The probability that the next event is a hang-up is,

$$\frac{3\gamma}{3\gamma + \lambda(9-3)} = \frac{\gamma}{\gamma + 2\lambda} \tag{9}$$

The blocking probabilities for $2a2$ and $2b2$ are both 0. Their weighted average, therefore, is also 0. The blocking probabilities for $4a2$, $4b3$, $4c2$ and $4e3$ are $0, 2/5, 1/5$ and $2/5$, respectively. Their weighted average is $4/15$. Thus, the probability of blocking by moving down in the state hierarchy is 0 and by moving up it is $4/15$. We combine these using (8) and (9) and get the overall probability of blocking after one event followed by a call request,

$$\frac{4}{15}\frac{2\lambda}{\gamma + 2\lambda} + 0\frac{\gamma}{\gamma + 2\lambda} = \frac{8\lambda}{15(\gamma + 2\lambda)} \tag{10}$$

At this point, we consider two different cases: high traffic and low traffic systems. As noted, we may eliminate either one but not both of λ or γ by fixing our time unit appropriately such that the desired rate is forced to be equal to 1. Thus, when considering low traffic scenarios, we wish to eliminate γ. In this case, λ will be small meaning that higher order terms of λ become less significant. When considering high traffic scenarios, we wish to eliminate λ. In this case, γ becomes small and, similarly, higher order terms of γ become less significant.

For example, decision d_3 requires deciding between states $3b1$ and $3b2$. We find that the expected probabilities of blocking at first-order λ and γ are,

$$3b1 : \frac{\frac{1}{7}\gamma + \frac{47}{30}\lambda}{3\gamma + 6\lambda} \quad 3b2 : \frac{\frac{1}{7}\gamma + \frac{49}{30}\lambda}{3\gamma + 6\lambda} \tag{11}$$

Note that in either case, for high or low traffic,

$$\forall\lambda > 0 \, \forall\gamma > 0 \; \frac{\frac{1}{7}\gamma + \frac{47}{30}\lambda}{3\gamma + 6\lambda} < \frac{\frac{1}{7}\gamma + \frac{49}{30}\lambda}{3\gamma + 6\lambda} \tag{12}$$

Therefore, we choose $3b1$ over $3b2$. Most decisions in $K_{3,3}$ are such that the outcome is not affected by the rate of traffic.

We repeat this process for every state and resolve the decisions $\{d_1, d_3, d_{11}, d_{12}, d_{17}\}$. Note that d_{26} is not included in the list and still remains to be resolved; both first-order λ and γ blocking probabilities for d_{26} are equal.

To resolve d_{26}, we apply one additional step of this recursive method to the first-order λ and γ blocking probabilities to derive a second-order blocking probability. The process is identical except that instead of multiplying by the weighted average of the constant blocking probabilities, we multiply by the weighted average of the first-order λ and γ blocking probabilities. In this way, all decisions are resolved.

4 Analysis Results

4.1 The Right Decisions

Figure 27 in the appendix displays the full decision results for $K_{3,3}$ and figure 30 displays those for K_5. One must keep in mind that for $K_{2,2}$, we found the actual globally optimal policy through enumeration of all policies. For $K_{3,3}$, we found an approximation to a globally optimal policy to within second-order λ or γ.

4.2 Emerging Global Patterns

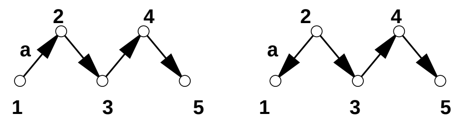

Figure 21. single chain decision

In all decisions examined, whenever a single chain appears as the only connected component and the edge placing the request lies on the end of the chain, the optimal decision choses the server towards the chain. For example, figure 21 displays two chains of length five. When client a makes a call request, we may choose between entering these two states. In this example, we would choose server 2 over server 1. Note that this is not the case if there are additional disconnected components within the graph.

4.3 Interesting Decisions

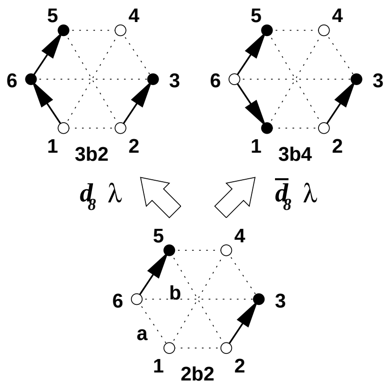

Figure 22. unexpected decision outcome

Contrary to possible conjecture, a call request placed by a client adjacent to a busy client does not always result in service being provided by the server common to the two. For example, figure 22 displays client a requesting service from either servers 1 or 6. We may guess that $3b2$ is advantageous when compared to $3b4$, since in $3b2$, server 1 has two idle neighbouring clients, whereas server 6 only has one in $3b4$. The better choice, however, is $3b4$. The explanation is straightforward; no clients are blocked in $3b4$, whereas client b is blocked in $3b2$.

4.4 Dependence upon Rate of Traffic

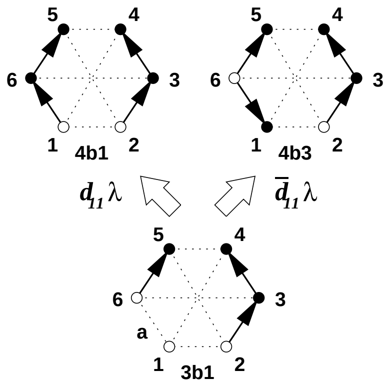

Figure 23. d_{11}: decision depends on λ

In deciding between two states, we compared blocking probabilities in constant, first and second-order λ and γ. We need to consider both high traffic and low traffic cases when comparing two expectation expressions. As mentionned, these do not always coincide.

Decision d_{11} involves a transition from $3b1 \rightarrow 4b1$ or from $3b1 \rightarrow 4b3$. The first-order blocking probabilities of $4b1$ and $4b3$ are,

$$4b1 : \frac{\frac{2}{3}\gamma + \frac{51}{20}\lambda}{4\gamma + 5\lambda} \quad 4b3 : \frac{\frac{1}{2}\gamma + \frac{14}{5}\lambda}{4\gamma + 5\lambda} \qquad (13)$$

The blocking probability for $4b1$ is preferable to that of $4b3$ when,

$$\frac{\frac{2}{3}\gamma + \frac{51}{20}\lambda}{4\gamma + 5\lambda} < \frac{\frac{1}{2}\gamma + \frac{14}{5}\lambda}{4\gamma + 5\lambda}$$

$$\Leftrightarrow \frac{2}{3}\gamma + \frac{51}{20}\lambda < \frac{1}{2}\gamma + \frac{14}{5}\lambda$$

$$\Leftrightarrow \left(\frac{2}{3} - \frac{1}{2}\right)\gamma < \left(\frac{14}{5} - \frac{51}{20}\right)\lambda$$

$$\Leftrightarrow \frac{2}{3}\gamma < \lambda \qquad (14)$$

Note that for $0 < \frac{2}{3}\gamma < \lambda$, $4b1$ has lower blocking probability than $4b3$, whereas, for $0 < \lambda < \frac{2}{3}\gamma$, the opposite is true.

Contrary to Beneš' conjecture, our conclusions about d_{11} demonstrate that a policy's optimality depends upon the rate of traffic. A decision may be optimal within some range of call request and hang-up rates but sub-optimal outside of that range. Interestingly, d_{11} in $K_{3,3}$ was the only such decision found out of all decisions examined for the network graphs K_3, K_4, K_5, $K_{2,2}$ and $K_{3,3}$. Presumably, other such decisions exist in more complex graphs.

4.5 Comparing Theoretical and Computational Results

As a quick test, we take the derived policy for $K_{3,3}$ and independently test single decisions. We take each decision one at a time, reverse it, calculate the the expected number of busy servers for the altered policy, v'_d, using equation (4) and compare the result with the original policy, v_d. We find that reversing any one of the 27 decisions, with the exception of d_{11}, results in $v'_d < v_d$ for all rates of traffic.

5 Future Work and Open Problems

Future considerations include possible attempts at solving the policies for the cube, the Petersen graph, the octahedron, the dodecahedron and $K_{4,4}$ which, of course, may require new analysis techniques or improvements on these techniques presented here. We may also require automation of the equivalence class generation, transition rate computation and decision analysis.

We may examine the significance of client-server ratio. In our models, we always considered clients that were connected to only two servers, which allowed us to use a graphical model where edges represented clients. Without this restriction, we might discover interesting results by allowing greater than two servers per client.

Finally, we may consider the existence of general decision rules to encompass families of network models such as K_n or $K_{n,n}$. Such a guide would allow general routing decisions to be made without having to perform analysis specific to the network.

References

[Arm88] M. A. Armstrong. *Groups and Symmetry*. Springer-Verlag, 1988.

[Ben66] V. E. Beneš. Programming and control problems arising from optimal routing in telephone networks. *Bell System Technical Journal*, 45(9):1373–1438, 1966.

[Ben78] V. E. Beneš. Reduction of network states under symmetries. *Bell System Technical Journal*, 57(1):111–149, 1978.

[SS65] Ya. Ya. Sedol and M. A. Shneps. Some quality investigations of partial access circuits. *Problemy Peredachi Informatsii*, 1(2):87–94, 1965.

6 Appendix

1. Figure 24 shows busy server states and busy client states for $K_{3,3}$ (vertex permutations and connected edge components which can be embedded into the graph, respectively).

2. Figure 25 displays all client-server equivalence classes for $K_{3,3}$.

3. Tables A1 and A2 give all forward and backward transition rates between states in $K_{3,3}$.

4. Figure 27 display complete decisions results for $K_{3,3}$.

5. Figure 28 shows the client-server equivalence classes for K_5.

6. Tables A3 and A4 give all forward and backward transition rates between states in K_5.

7. Figure 30 displays complete decisions results for K_5.

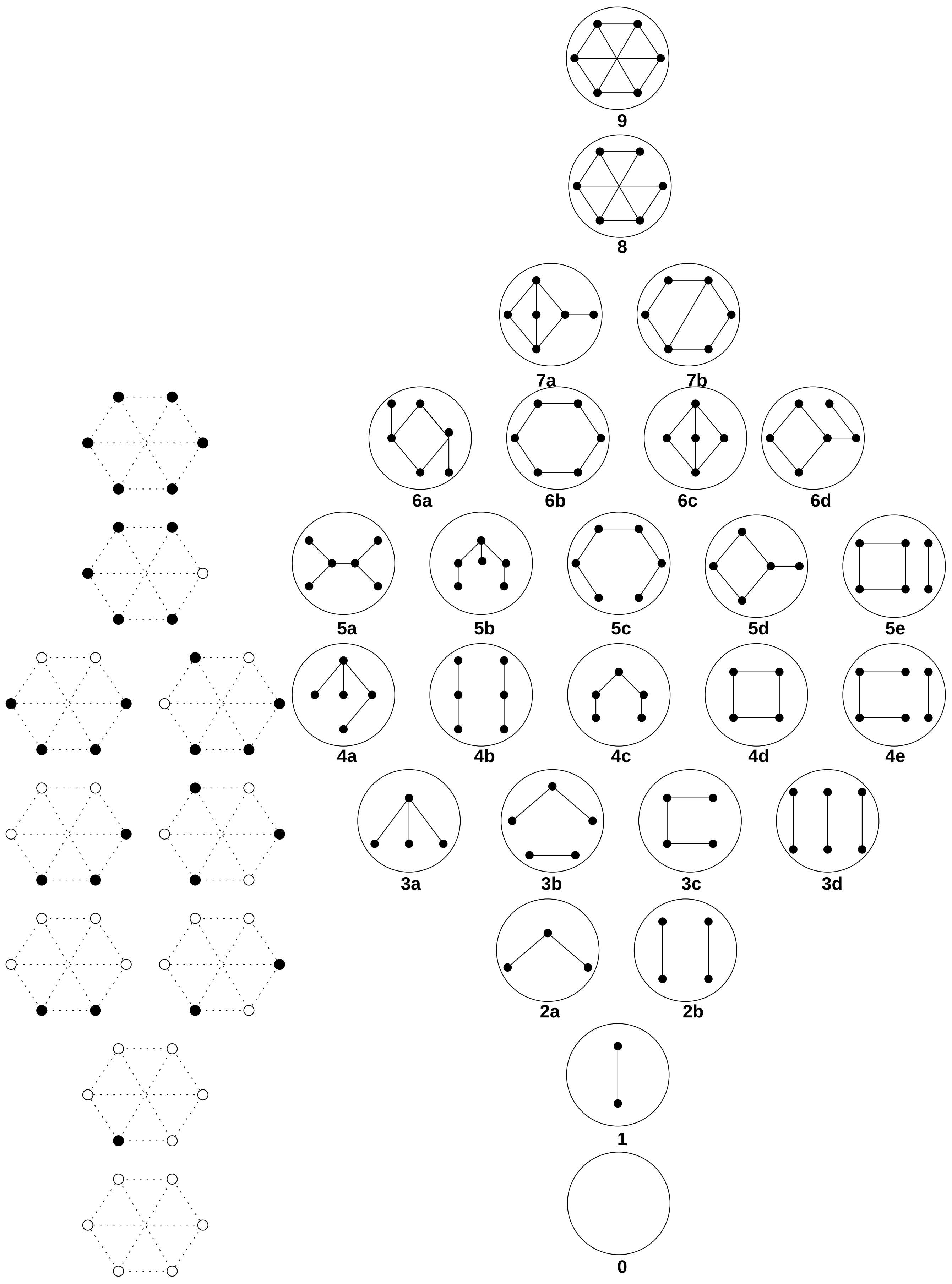

Figure 24. busy server states and busy client states for $K_{3,3}$

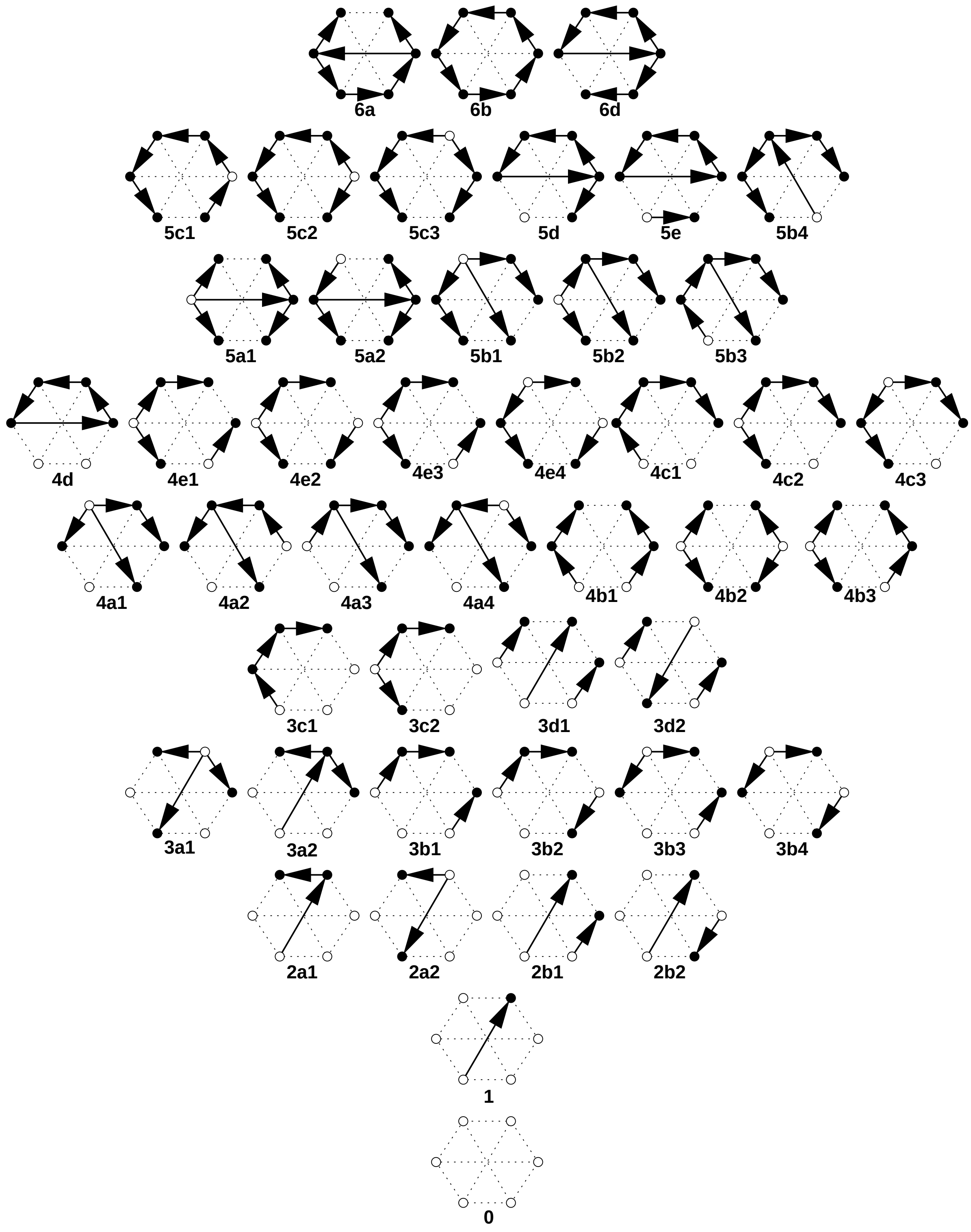

Figure 25. client-server states for $K_{3,3}$

Table A1

State			Forward Transition Rates for $K_{3,3}$						
1	0	$\rightarrow$	1 9λ						
2	1	$\rightarrow$	2a1 $(2+2d_1)\lambda$	2a2 $2d_1\lambda$	2b1 $4d_2\lambda$	2b2 $4d_2\lambda$			
3	2a1	$\rightarrow$	3a2 λ	3b1 $2d_3\lambda$	3b2 $2d_3\lambda$	3c1 $(2+2d_4)\lambda$	3c2 $2d_4\lambda$		
4	2a2	$\rightarrow$	3a1 $d_5\lambda$	3a2 $\bar{d}_5\lambda$	3b3 $2d_6\lambda$	3b4 $2\bar{d}_6\lambda$	3c2 4λ		
5	2b1	$\rightarrow$	3b1 $2d_7\lambda$	3b2 2λ	3b3 $2\bar{d}_7\lambda$	3c2 λ	3d1 λ		
6	2b2	$\rightarrow$	3b1 2λ	3b2 $2d_8\lambda$	3b4 $2\bar{d}_8\lambda$	3c1 2λ	3d1 $d_9\lambda$	3d2 $\bar{d}_9\lambda$	
7	3a1	$\rightarrow$	4a1 6λ						
8	3a2	$\rightarrow$	4a2 $2d_{10}\lambda$	4a3 4λ	4a4 $2\bar{d}_{10}\lambda$				
9	3b1	$\rightarrow$	4a3 λ	4b1 $d_{11}\lambda$	4b3 $\bar{d}_{11}\lambda$	4e1 $d_{12}\lambda$	4e3 $\bar{d}_{12}\lambda$	4e2 λ	4c1 λ
10	3b2	$\rightarrow$	4b1 λ	4c1 λ	4c2 $d_{13}\lambda$	4c3 $\bar{d}_{13}\lambda$	4e1 λ	4e2 $d_{14}\lambda$	4e4 $\bar{d}_{14}\lambda$
11	3b3	$\rightarrow$	4a1 $d_{15}\lambda$	4a4 $\bar{d}_{15}\lambda$	4b2 $d_{16}\lambda$	4b3 $\bar{d}_{16}\lambda$	4e4 2λ		
12	3b4	$\rightarrow$	4a2 λ	4b3 λ	4c2 2λ	4e3 2λ			
13	3c1	$\rightarrow$	4a2 λ	4a3 λ	4c1 $(1+d_{17})\lambda$	4c2 $\bar{d}_{17}\lambda$	4d λ	4e1 $d_{18}\lambda$	4e2 $\bar{d}_{18}\lambda$
14	3c2	$\rightarrow$	4a1 $d_{19}\lambda$	4a3 $\bar{d}_{19}\lambda$	4a4 λ	4c2 λ	4c3 λ	4e3 $d_{20}\lambda$	4e4 $\bar{d}_{20}\lambda$
15	3d1	$\rightarrow$	4e1 2λ	4e3 $2d_{21}\lambda$	4e4 $2\bar{d}_{21}\lambda$				
16	3d2	$\rightarrow$	4e2 6λ						
17	4a1	$\rightarrow$	5a1 λ	5b1 2λ					
18	4a2	$\rightarrow$	5a2 λ	5b3 2λ	5d 2λ				
19	4a3	$\rightarrow$	5a2 λ	5b2 $d_{22}\lambda$	5b3 $\bar{d}_{22}\lambda$	5b4 λ	5d λ		
20	4a4	$\rightarrow$	5a1 $d_{23}\lambda$	5a2 $\bar{d}_{23}\lambda$	5b2 2λ				
21	4b1	$\rightarrow$	5c1 2λ	5c3 λ					
22	4b2	$\rightarrow$	5a1 λ						
23	4b3	$\rightarrow$	5a2 λ	5c2 2λ					
24	4c1	$\rightarrow$	5b3 λ	5c1 $(1+d_{24})\lambda$	5c2 $\bar{d}_{24}\lambda$	5d λ			
25	4c2	$\rightarrow$	5b2 λ	5c2 λ	5c3 λ	5d λ			
26	4c3	$\rightarrow$	5b1 $d_{25}\lambda$	5b4 $\bar{d}_{25}\lambda$	5c3 2λ				
27	4d	$\rightarrow$	5d 4λ	5e λ					
28	4e1	$\rightarrow$	5b3 λ	5c2 $d_{26}\lambda$	5c3 $\bar{d}_{26}\lambda$	5e λ			
29	4e2	$\rightarrow$	5b4 λ	5c1 2λ	5e λ				
30	4e3	$\rightarrow$	5b2 λ	5b3 λ	5c3 λ				
31	4e4	$\rightarrow$	5b1 $d_{27}\lambda$	5b2 $\bar{d}_{27}\lambda$	5c2 λ				
32	5a1	$\rightarrow$	null						
33	5a2	$\rightarrow$	6a 2λ						
34	5b1	$\rightarrow$	null						
35	5b2	$\rightarrow$	6a λ						
36	5b3	$\rightarrow$	6a λ	6d λ					
37	5b4	$\rightarrow$	6d 2λ						
38	5c1	$\rightarrow$	6b λ	6d λ					
39	5c2	$\rightarrow$	6a λ						
40	5c3	$\rightarrow$	6d λ						
41	5d	$\rightarrow$	6a 2λ	6d λ					
42	5e	$\rightarrow$	6d 2λ						
43	6a	$\rightarrow$	null						
44	6b	$\rightarrow$	null						
45	6d	$\rightarrow$	null						

Table A2

State			Backward Transition Rate for $K_{3,3}$					
1	0	$\rightarrow$	null					
2	1	$\rightarrow$	0 γ					
3	2a1	$\rightarrow$	1 2γ					
4	2a2	$\rightarrow$	1 2γ					
5	2b1	$\rightarrow$	1 2γ					
6	2b2	$\rightarrow$	1 2γ					
7	3a1	$\rightarrow$	2a2 3γ					
8	3a2	$\rightarrow$	2a1 2γ	2a2 γ				
9	3b1	$\rightarrow$	2a1 γ	2b1 γ	2b2 γ			
10	3b2	$\rightarrow$	2a1 γ	2b1 γ	2b2 γ			
11	3b3	$\rightarrow$	2a2 γ	2b1 2γ				
12	3b4	$\rightarrow$	2a2 γ	2b2 2γ				
13	3c1	$\rightarrow$	2a1 2γ	2b2 γ				
14	3c2	$\rightarrow$	2a1 γ	2a2 γ	2b1 γ			
15	3d1	$\rightarrow$	2b1 2γ	2b2 γ				
16	3d2	$\rightarrow$	2b2 3γ					
17	4a1	$\rightarrow$	3a1 γ	3b3 γ	3c2 2γ			
18	4a2	$\rightarrow$	3a2 γ	3b4 γ	3c1 2γ			
19	4a3	$\rightarrow$	3a2 γ	3b1 γ	3c1 γ	3c2 γ		
20	4a4	$\rightarrow$	3a2 γ	3b3 γ	3c2 2γ			
21	4b1	$\rightarrow$	3b1 2γ	3b2 2γ				
22	4b2	$\rightarrow$	3b3 4γ					
23	4b3	$\rightarrow$	3b1 γ	3b3 γ	3b4 2γ			
24	4c1	$\rightarrow$	3b1 γ	3b2 γ	3c1 2γ			
25	4c2	$\rightarrow$	3b2 γ	3b4 γ	3c1 γ	3c2 γ		
26	4c3	$\rightarrow$	3b2 2γ	3c2 2γ				
27	4d	$\rightarrow$	3c1 4γ					
28	4e1	$\rightarrow$	3b1 γ	3b2 γ	3c1 γ	3d1 γ		
29	4e2	$\rightarrow$	3b1 γ	3b2 γ	3c1 γ	3d2 γ		
30	4e3	$\rightarrow$	3b1 γ	3b4 γ	3c2 γ	3d1 γ		
31	4e4	$\rightarrow$	3b2 γ	3b3 γ	3c2 γ	3d1 γ		
32	5a1	$\rightarrow$	4a1 2γ	4a4 2γ	4b2 γ			
33	5a2	$\rightarrow$	4a2 γ	4a3 2γ	4a4 γ	4b3 γ		
34	5b1	$\rightarrow$	4a1 2γ	4c3 γ	4e4 2γ			
35	5b2	$\rightarrow$	4a3 γ	4a4 γ	4c2 γ	4e3 γ	4e4 γ	
36	5b3	$\rightarrow$	4a2 γ	4a3 γ	4c1 γ	4e1 γ	4e3 γ	
37	5b4	$\rightarrow$	4a3 2γ	4c3 γ	4e2 2γ			
38	5c1	$\rightarrow$	4b1 γ	4c1 2γ	4e2 2γ			
39	5c2	$\rightarrow$	4b3 γ	4c1 γ	4c2 γ	4e1 γ	4e4 γ	
40	5c3	$\rightarrow$	4b1 γ	4c2 γ	4c3 γ	4e1 γ	4e3 γ	
41	5d	$\rightarrow$	4a2 γ	4a3 γ	4c1 γ	4c2 γ	4d γ	
42	5e	$\rightarrow$	4d γ	4e1 2γ	4e2 2γ			
43	6a	$\rightarrow$	5a2 γ	5b2 γ	5b3 γ	5c2 γ	5d 2γ	
44	6b	$\rightarrow$	5c1 6γ					
45	6d	$\rightarrow$	5b3 γ	5b4 γ	5c1 γ	5c3 γ	5d γ	5e γ

Figure 26. transition rates for $K_{3,3}$

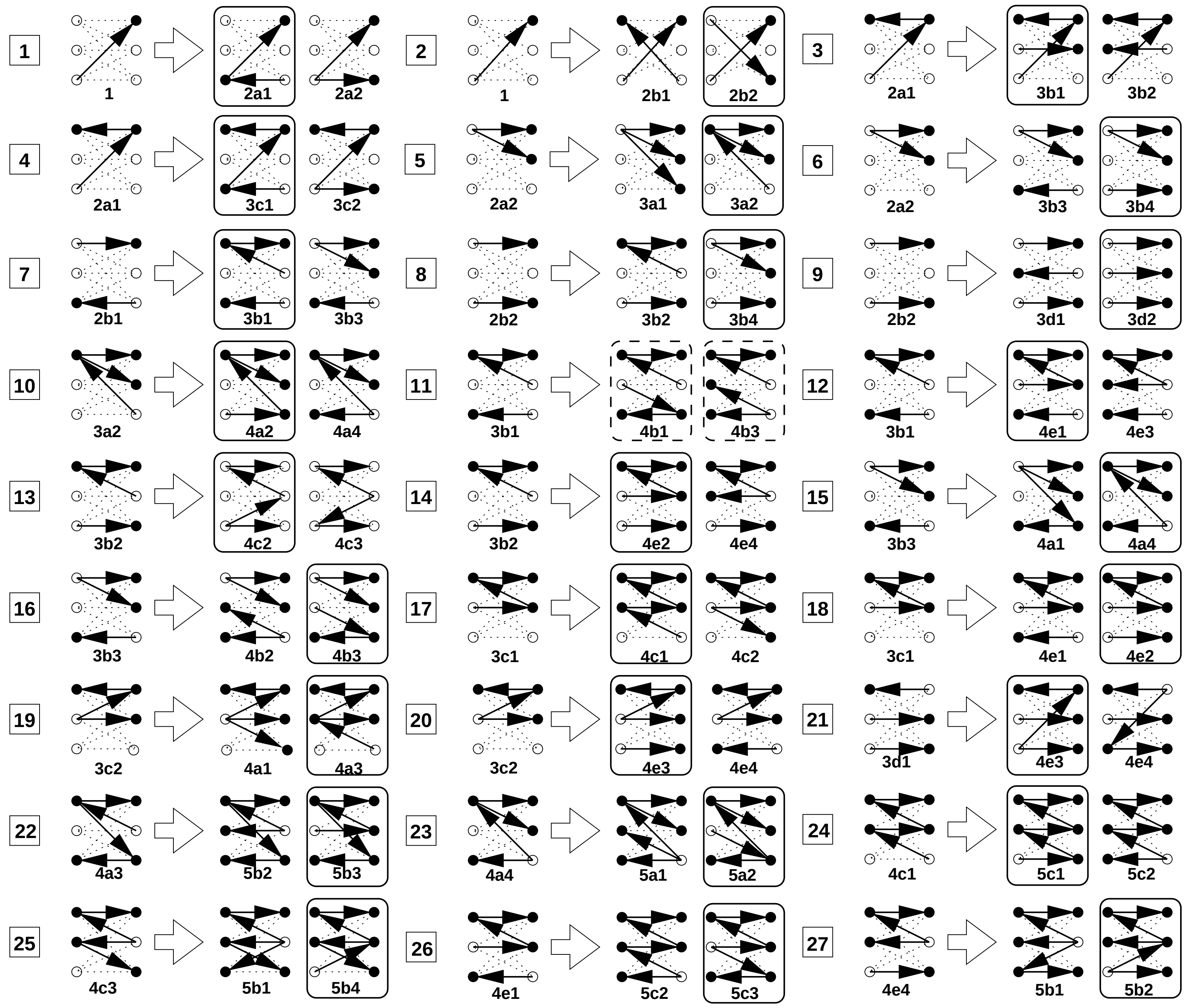

Figure 27. optimal decisions for $K_{3,3}$

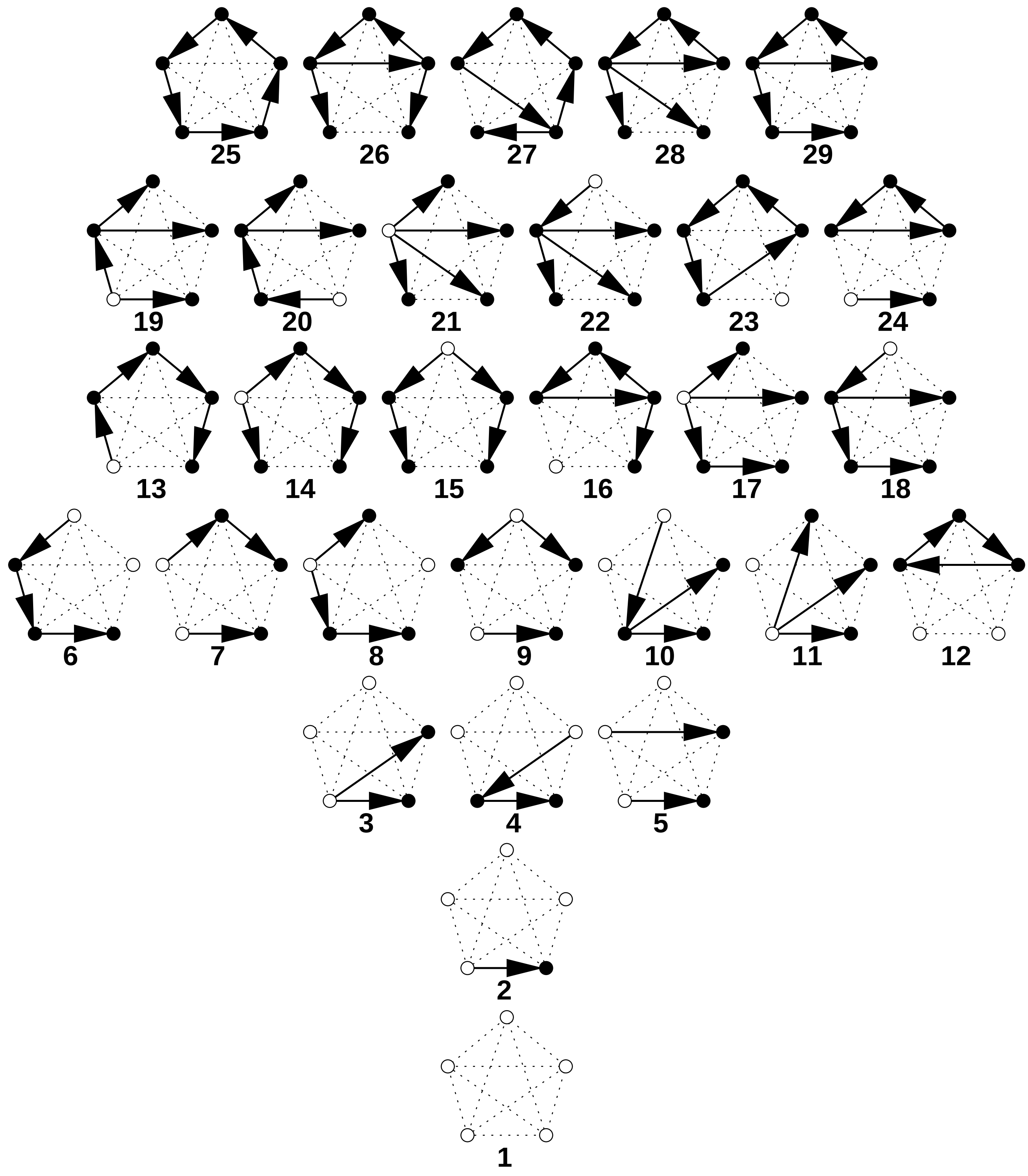

Figure 28. client-server states for K_5

Table A3

State		Forward Transition Rates for K_5					
1	$\rightarrow$	10λ					
2	$\rightarrow$	3 $3d_1\lambda$	4 $(3+3d_1)\lambda$	5 3λ			
3	$\rightarrow$	8 4λ	9 λ	10 $2d_2\lambda$	11 $2d_2\lambda$		
4	$\rightarrow$	6 $(2+2d_3)\lambda$	7 λ	8 $2\bar{d}_3\lambda$	10 2λ	12 λ	
5	$\rightarrow$	6 2λ	7 $(2+2d_4)\lambda$	8 λ	9 $2\bar{d}_4\lambda$		
6	$\rightarrow$	13 $(1+d_5)\lambda$	14 $d_5\lambda$	16 λ	18 λ	20 λ	23 λ
7	$\rightarrow$	13 2λ	14 $d_6\lambda$	15 $\bar{d}_6\lambda$	18 λ	24 λ	
8	$\rightarrow$	14 λ	15 λ	16 λ	17 $d_7\lambda$	18 $\bar{d}_7\lambda$	19 λ
9	$\rightarrow$	14 2λ	17 $d_8\lambda$	19 $\bar{d}_8\lambda$	20 λ		
10	$\rightarrow$	16 2λ	18 2λ	19 $d_9\lambda$	20 $\bar{d}_9\lambda$	22 λ	
11	$\rightarrow$	17 3λ	21 $d_{10}\lambda$	22 $\bar{d}_{10}\lambda$			
12	$\rightarrow$	16 6λ	24 λ				
13	$\rightarrow$	25 λ	27 λ	29 λ			
14	$\rightarrow$	26 λ	27 λ				
15	$\rightarrow$	29 2λ					
16	$\rightarrow$	26 2λ	28 λ	29 λ			
17	$\rightarrow$	28 λ					
18	$\rightarrow$	26 λ	27 λ	29 λ			
19	$\rightarrow$	26 2λ					
20	$\rightarrow$	27 2λ	28 λ				
21	$\rightarrow$	null					
22	$\rightarrow$	28 3λ					
23	$\rightarrow$	27 4λ					
24	$\rightarrow$	29 3λ					
25	$\rightarrow$	null					
26	$\rightarrow$	null					
27	$\rightarrow$	null					
28	$\rightarrow$	null					
29	$\rightarrow$	null					

Table A4

State		Backward Transition Rates for K_5				
1	$\rightarrow$	null				
2	$\rightarrow$	1 γ				
3	$\rightarrow$	2 2γ				
4	$\rightarrow$	2 2γ				
5	$\rightarrow$	2 2γ				
6	$\rightarrow$	4 2γ	5 γ			
7	$\rightarrow$	4 γ	5 2γ			
8	$\rightarrow$	3 γ	4 γ	5γ		
9	$\rightarrow$	3 γ	5 2γ			
10	$\rightarrow$	3 γ	4 2γ			
11	$\rightarrow$	3 3γ				
12	$\rightarrow$	4 3γ				
13	$\rightarrow$	6 2γ	7 2γ			
14	$\rightarrow$	6 γ	7 γ	8 γ	9 γ	
15	$\rightarrow$	7 2γ	8 2γ			
16	$\rightarrow$	6 γ	8 γ	10 γ	12 γ	
17	$\rightarrow$	8 2γ	9 γ	11 γ		
18	$\rightarrow$	6 γ	7 γ	8 γ	10 γ	
19	$\rightarrow$	8 2γ	9 γ	10 γ		
20	$\rightarrow$	6 2γ	9 γ	10 γ		
21	$\rightarrow$	11 4γ				
22	$\rightarrow$	10 3γ	11 γ			
23	$\rightarrow$	6 4γ				
24	$\rightarrow$	7 3γ	12 γ			
25	$\rightarrow$	13 5γ				
26	$\rightarrow$	14 γ	16 2γ	18 γ	19 γ	
27	$\rightarrow$	13 γ	14 γ	18 γ	20 γ	23 γ
28	$\rightarrow$	16 2γ	17 γ	20 γ	22 γ	
29	$\rightarrow$	13 γ	15 γ	16 γ	18 γ	24 γ

Figure 29. transition rates for K_5

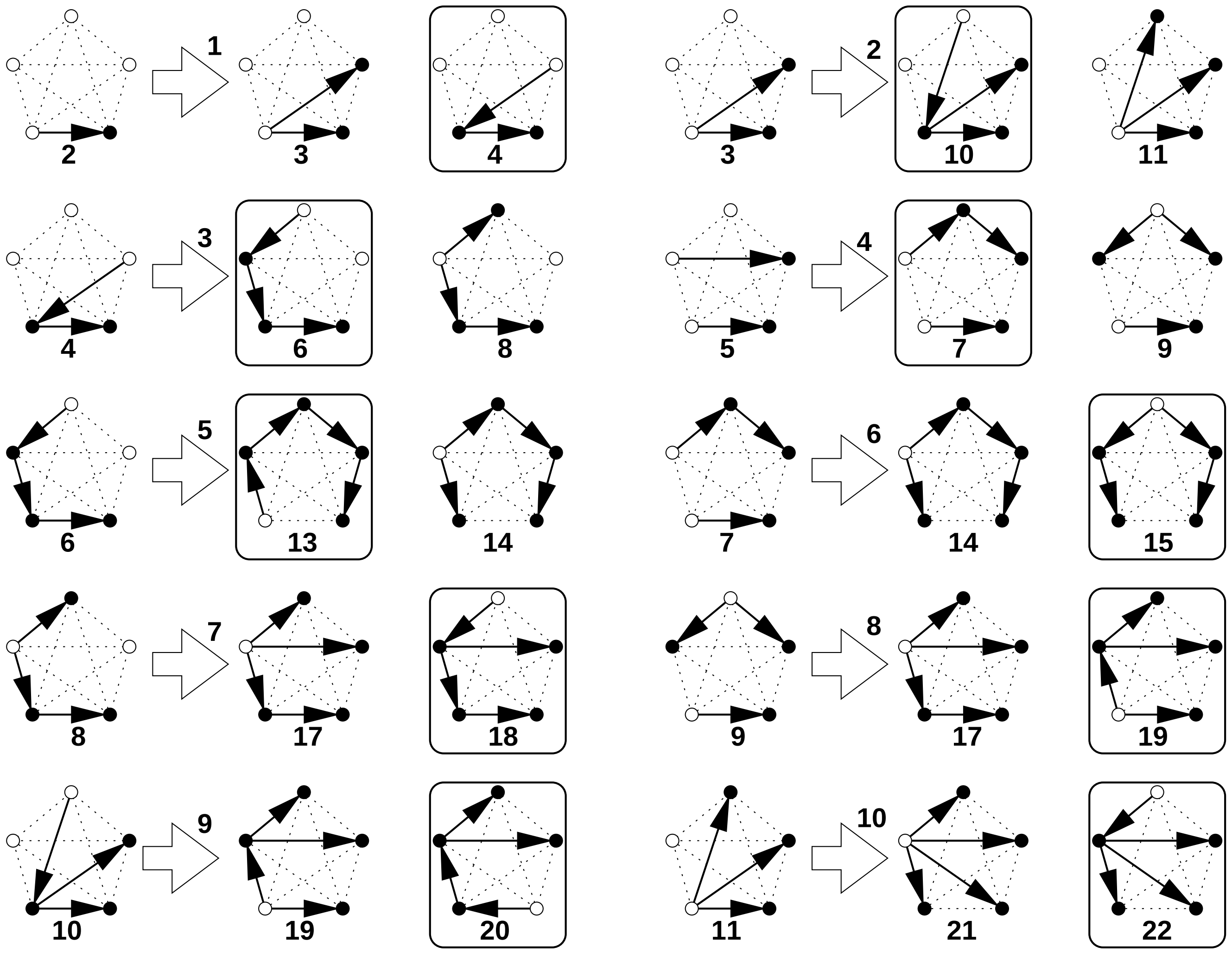

Figure 30. optimal decisions for K_5